ROKEY BOOT CAMP

ROS2 기초-1차시

Apr, 2025

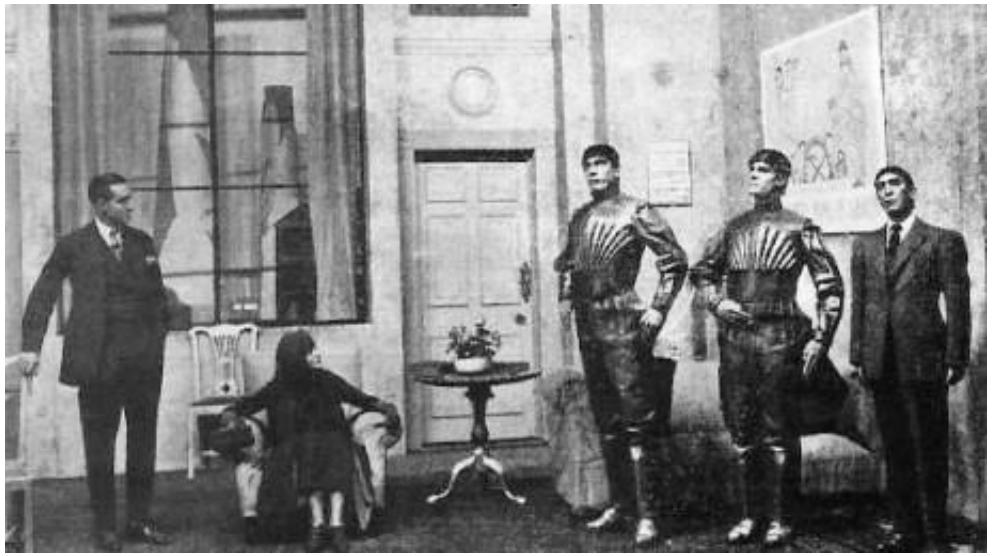
훈련 일정

	오전	오후
1차시	▪ 로봇의 역사 ▪ 컴퓨터 구조(Booting, CPU 작동원리, POST)	▪ 리눅스와 운영체계 ▪ 리눅스 CLI 실습(디렉토리, 계정, 기본 명령어등), Terminator, 커널, 쉘, gedit, bash
	▪ Application 작동원리(마이크로 프로세서, 메모리, 저장장치)	▪ 리눅스 CLI 실습
2차시	▪ 리눅스 CLI 실습	▪ 네트워크와 통신(IPV4, IPV6, 소켓, 노드, Hub, TCP, UDP)
	▪ API, Library, Framework, 프로세스와 Thread ▪ 인터프리터, 컴파일러(소스코드 → Build → 실행파일)	▪ 소켓프로그래밍 실습 ▪ OSI7 Layer, 프로토콜(RS-232C, RS-485, Ethernet, TCP/UDP, IP)
3차시	▪ 센서 기초, IoT와 Embedded	▪ 로봇기초, 좌표계
	▪ 로봇 센서 활용 및 로봇의 구성(기계기구, 전기전자, 소프트웨어)	▪ ROS2 소개 및 활용 ▪ ROS2 설치(ros.org) 및 demo_node
4차시	▪ ROS2 소개 및 활용	▪ ROS2 실습(Talker, Listener)
	▪ ROS2 패키지 설명	▪ ROS2 실습(Turtlesim, teleop_key)
5차시	▪ ROS2 실습(Turtlesim, Teleop_key 여러 개 만들기)	▪ Topic, Service, Action, Parameter, RQT, RQT_Graph 이론 및 실습
	▪ ROS2 실습(Turtlesim, Namespace 여러 개 만들기)	▪ Ros bag and play 실습, my first package build 실습 ▪ Turtlesim subscribing 실습, ROS의 중요한 개발도구(Rviz, GAZEBO 소개)

로봇의 어원

로봇(Robot)은 어떤 작업이나 조작을 자동으로 할 수 있는 기계장치를 말합니다.
오늘날 우리가 익숙하게 쓰는 '로봇'이라는 용어는 1920년에 체코슬로바키아 극작가인 차펙(Karel Čapek)이 <로섬의 만능로봇(Rossum's Universal Robots, R.U.R.)>이라는 희곡에서 처음 사용한 것이 그 기원이 되었다고 합니다.

로봇의 어원은 체코어로 천한 노동, 중노동, 강제노동 등을 뜻하는 '로보타(robota)'인데요. 연극의 내용은 뛰어난 과학자 로섬과 그의 아들이 인간에게 무조건 복종하고 모든 육체적 노동을 대신해 줄 로봇을 만들어내는데 로섬의 동료가 로봇에게 감정을 준 이후 로봇이 점점 일을 싫어하게 되면서 결국은 반란을 일으켜 사람들을 죽이고 세계를 정복하게 된다는 내용입니다. 영화 <터미네이터>, <매트릭스> 등 여러 SF영화의 줄거리가 떠오르기도 합니다. 그만큼 극작가 차펙의 희곡이 후대에 끼친 영향이 크다는 것이겠죠? 물론 이때까지도 로봇은 이렇게 상상 속의 존재에 불과했습니다.



[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

진짜 로봇의 탄생

상상 속의 존재였던 로봇. 그러다 1950년대에 최초의 산업용 로봇 유니메이트가 등장하면서 로봇에 관한 연구가 급속도로 발전하기 시작합니다. 최초의 산업용 로봇 유니메이트는 공장에서 막 생산되어 뜨거운 금속 사출물을 운반하는 역할을 했습니다. 인간이 하기 힘든 일을 대신하기 시작한 것이죠.

유니메이트 이후에도 장애인 환자를 돕는 로봇팔, 자동차를 조립하는 생산용 로봇 등이 나오기 시작했습니다. 단순 반복 작업이 가능하고 부상의 위험이 없다는 점 때문에 다양한 분야에서 생산 공정에 맞는 로봇을 만들어 사용하였죠.

용도와 사용처가 명확하기 때문에 이러한 산업용 로봇들은 인간을 닮기보다는 굴삭기나 재봉틀처럼 용도에 딱 맞는 기계에 더 가까운 모습을 하고 있었습니다.

1980년대 이후에는 용도에 따라 산업용, 의료용, 가정용, 탐사용, 군사용 등 그 용도가 다양해졌습니다. 로봇을 전문적으로 제작하는 기업도 생겼고 연구자도 늘어났습니다. 연구가 점점 고도화되면서 로봇을 조종하는 방법도 용도에 따라 달라졌고요. 그 결과 인간이 직접 수동조작을 해야 하는 로봇, 인간이 설정해둔 순서대로 따라하는 로봇을 넘어 스스로 학습 능력이나 판단력을 지닌 로봇까지 개발되면서 발전을 거듭하게 되었습니다.

[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

로봇의 역사

로봇의 발전

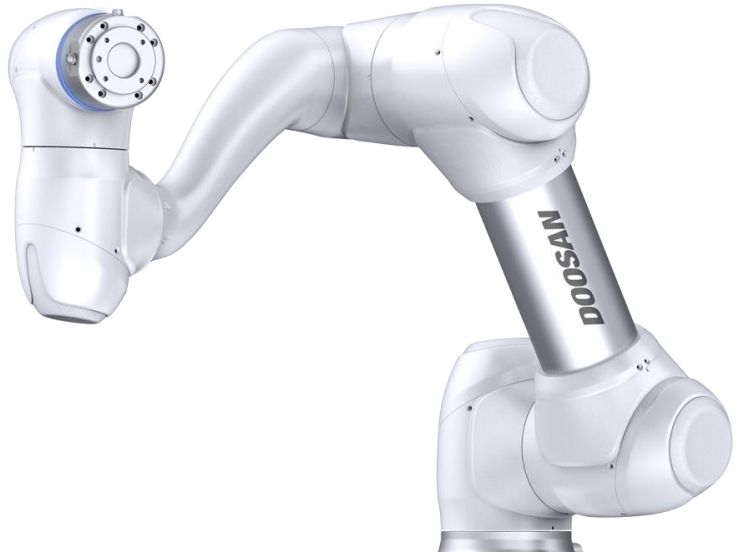
1999년 일본에서 출시된 반려견을 닮은 강아지 로봇 아이보(AiBo)는 실제 강아지처럼 인간과 상호 교류가 가능해서 큰 인기를 끌었습니다.
그리고 마침내 2000년대에 이르러서는 한국과학기술연구원(KIST) 연구팀이 개발한 마루(MAHRU), 한국과학기술원(KAIST) 연구팀이 개발한 휴보(HUBO), 일본 혼다에서 개발한 아시모(ASIMO) 등 직립 보행하는 인간형 로봇이 만들어지기도 했습니다.
마루의 경우, 무선 네트워크로 연결된 서버 컴퓨터를 통해 로봇에게 인식 능력을 제공하는 '네트워크 기반 휴머노이드'로 개발되었다고 합니다. 휴보의 경우, 2013년 미국방부에서 열린 인명구조로봇 선발 대회인 DARPA 로보틱스 챌린지(DRC)에서 1위를 했다고 합니다. 당시 대회에서는 차량 운전, 사다리 타기, 장애물 제거 등 인간 구조대원이 실제로 재난 현장에서 하는 활동을 해내야 했다고 하는데 이 대회에서 1등을 했다니 대단하죠?

[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

로봇의 역사

로봇의 발전

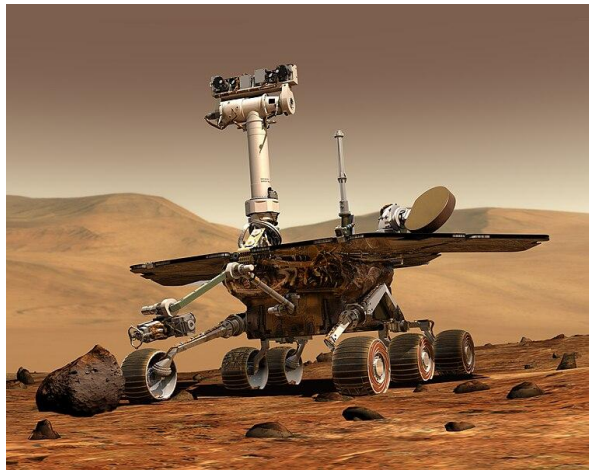
이처럼 로봇 관련 연구는 소수의 산업용 로봇으로 시작하여 보다 정확하고 안전한 수술을 할 수 있도록 돕는 의료용 로봇이나 인간과 대화하며 상호작용을 하는 대화형 로봇에 이르기까지 꾸준히 발전해 왔습니다. 그러다 이제는 집안을 스스로 청소하는 로봇청소기처럼 가정용으로 보급되기까지에 이르렀습니다. 산업용 로봇이 처음 출시된 게 1950년대였던 걸 생각하면 약 70여년 동안 이루어진 발전이 어마무시하다는 생각이 듭니다.



[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

로봇의 미래

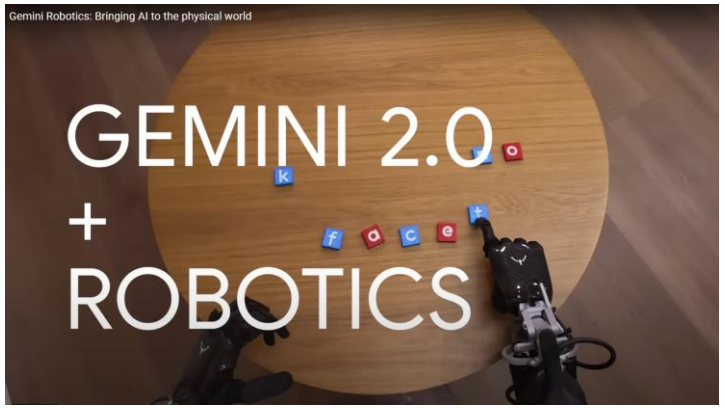
그렇다면 앞으로 로봇은 어떻게 발전하게 될까요?
불과 몇 년 전 있었던 코로나19의 전 세계적인 유행으로 인해 다양한 분야에서 비대면 수요가 크게 증가했던 걸 다들 기억하실 겁니다. 당시 전염병 위험 때문에 모든 것이 마비되고 멈췄다고 해도 과언이 아닌데요. 로봇은 전염병에 걸리지도 않는데다 전파할 위험도 없습니다. 게다가 쉬지 않고 일을 할 수 있기 때문에 생산작업 분야에서 수요가 크게 증가하면서 사회 곳곳에 빠르게 자리잡게 되었습니다.

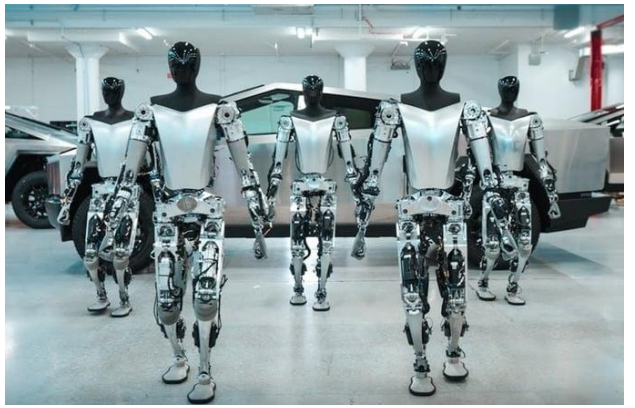


[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

로봇의 미래

코로나가 잠잠해진 요즘도 단순하고 반복적인 노동이 필요한 분야 또는 인간이 하기에 신체적 부상의 위험이 있는 분야를 중심으로 로봇이 적극 사용되는 중입니다. 그래서 우리는 익숙하게 키오스크로 음식을 주문하고 있습니다. 로봇은 단순히 주문을 받는 것뿐만 아니라 사람 대신 음식이나 음료를 만들기도 하고 서빙을 하기도 합니다. 덕분에 키오스크로 주문을 하고 로봇이 음료를 만드는 카페의 경우에는 아예 직원없이 무인으로 운영되는 것도 가능해졌죠.
공항이나 박물관 등에는 여러 가지 언어로 사람들의 질문에 친절히 답해주는 안내원 로봇이 있게 되었고요.
야간에는 사람 대신 공원을 순찰하는 순찰로봇이 위험을 방지합니다. 일손이 부족한 물류센터에서 빠른 일처리를 돕는 물류로봇이 등장하기도 했습니다. 우리가 자각하지 못할 만큼 자연스럽게 로봇이 일상생활 속으로 스며들고 있는 겁니다.





[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

로봇의 역사


We're partnering with Apptronik
to build the next generation of
humanoid robots with Gemini 2.0.


이 세상이 존재하기 위해선
다른 세상이 멸망해야 한다
트랜스포머
최후의 기사
FILMED WITH IMAX 3D CAMERAS
6월 21일 3D & 2D 대개봉


AI&DATA 기반 자동제어
AI기반 차량 부품 관리 지능형 제어시스템


YTN 사이언스
사이언스 투데이
'로봇 개'가 공장 지킨다…기아 광명공장 시범 운영

로봇의 역사

하지만 미래에 어떤 로봇이 만들어지든 아이작 아시모프(Isaac Asimov)라는 SF소설가가 만들었던 로봇이 지켜야 할 원칙은 지켜지게 될 거예요.
1942년 아이작 아시모프의 SF소설 '런어라운드(Runaround)'에서 처음 언급된 로봇의 3원칙은 다음과 같습니다.

제1원칙 **로봇은 인간에게 해를 끼쳐서는 안되며 위험에 처해있는 인간을 방관해서는 안 된다.**

제2원칙 **제1원칙에 위배되지 않는 경우 로봇은 인간의 명령에 반드시 복종해야만 한다.**

제3원칙 **제1원칙과 제2원칙에 위배되지 않는 경우 로봇은 자기 자신을 보호해야 한다.**

3원칙을 만든 이후 아이작 아시모프는 인류의 집단 안전을 위해 제0원칙을 추가했다고 합니다.
제0원칙은 다른 3원칙보다 더 상위에서 절대적으로 지켜야 하는 법칙이라고 해요.

제0원칙 **로봇은 인류에게 해를 가하거나 행동을 하지 않음으로써 인류에게 해가 가도록 해서는 안 된다.**

[출처] 로봇의 탄생과 발전 그리고 미래|**작성자** 한국과학기술연구원

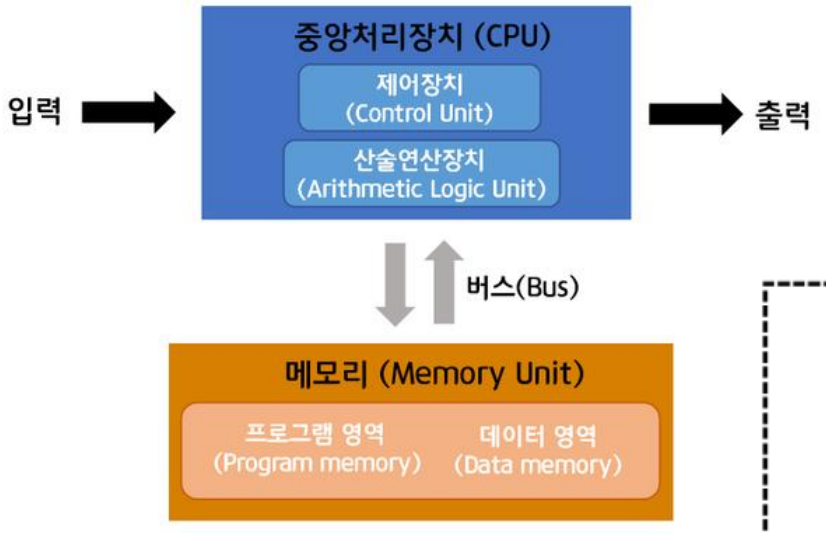
컴퓨터 구조(Booting, CPU 작동원리, POST)

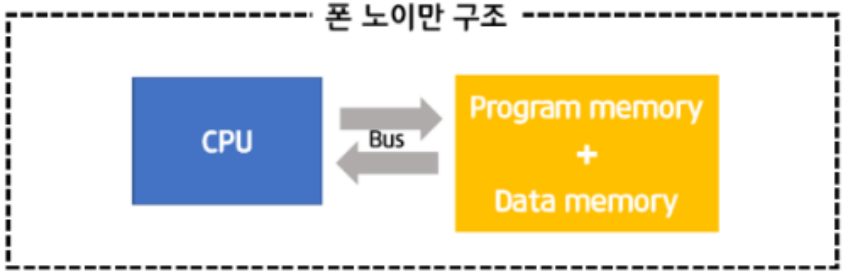
폰 노이만 구조

폰 노이만 구조는 중앙처리장치(CPU), 메모리, 프로그램 세 가지 요소로 구성되어 있습니다.

CPU와 메모리는 서로 분리되어 있고 둘을 연결하는 버스(Bus)를 통해 명령어 읽기, 데이터의 읽고 쓰기가 가능 -> 메모리 안에 프로그램과 데이터 영역의 물리적 구분 없음 -> **같은 버스를 통해 CPU가 명령어와 데이터에 동시 접근 불가**

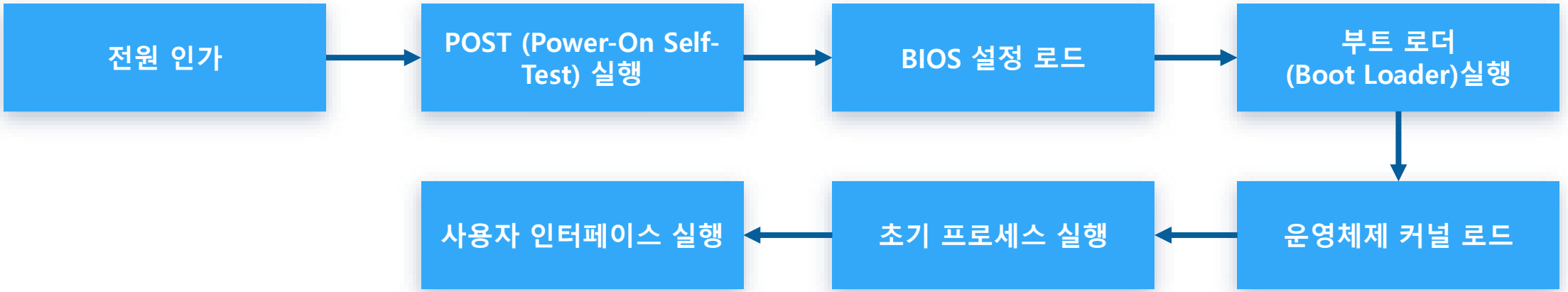
프로그램 내장 방식 컴퓨터 -> 폰 노이만 이전 하드웨어 전선을 바꿔야 함 -> 폰 노이만은 프로그램만 바꾸면 되기에 편의성이 증가



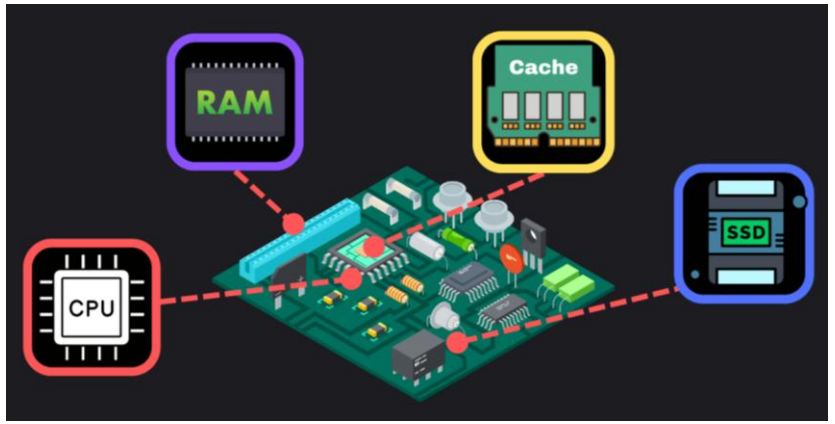


▲사진 5. 영화 <이미테이션 게임>에 나오는 하드웨어식 컴퓨터.

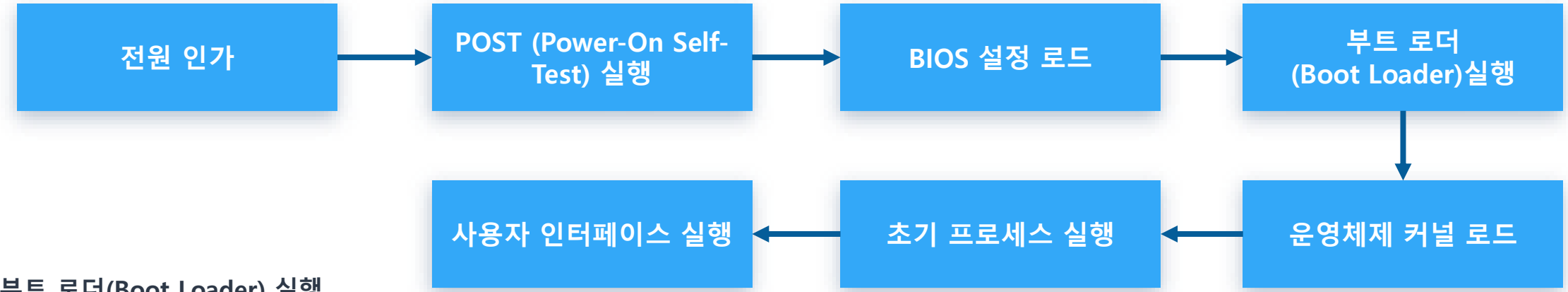
컴퓨터 구조(Booting, CPU 작동원리, POST)



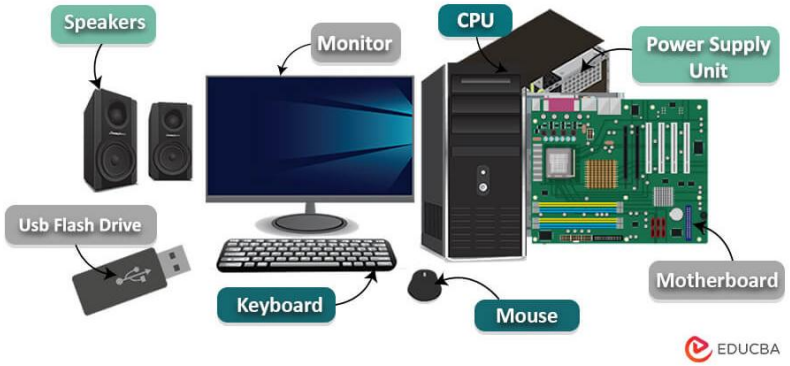
1. 전원 인가(Power On)
   - 컴퓨터의 전원 버튼을 누르면 메인보드에 전원이 공급되면서 부팅 과정이 시작됩니다. 메인보드의 BIOS(Basic Input/Output System) 칩에 저장된 펌웨어가 실행됩니다.
2. POST(Power-On Self-Test) 실행
   - BIOS는 컴퓨터의 주요 하드웨어 구성요소(CPU, 메모리, 그래픽카드 등)들을 점검하는 POST를 실행합니다. 이상이 없으면 짧은 비프음을 내고 다음 단계로 진행합니다.
3. BIOS 설정 로드
   - POST를 통과하면 BIOS는 CMOS(Complementary Metal-Oxide Semiconductor)에 저장된 설정값을 읽어옵니다. 여기에는 부팅 순서, 시간, 하드 디스크 정보 등이 포함됩니다.



컴퓨터 구조(Booting, CPU 작동원리, POST)



4. 부트 로더(Boot Loader) 실행
   - BIOS는 설정된 부팅 순서에 따라 부팅 가능한 장치(하드디스크, USB, CD-ROM 등)를 찾아 부트 로더를 실행합니다. 부트 로더는 운영체제 커널을 메모리에 적재하고 제어권을 넘기는 역할을 합니다.
5. 운영체제 커널 로드
   - 부트 로더는 하드디스크에서 운영체제 커널을 찾아 메모리에 적재합니다. 커널은 운영체제의 핵심 부분으로, 하드웨어를 제어하고 시스템 자원을 관리합니다.
6. 초기 프로세스 실행
   - 커널은 시스템 초기화를 마치고 init(리눅스) 또는 smss.exe(윈도우) 같은 첫 번째 프로세스를 실행합니다. 이 프로세스는 운영체제의 나머지 부분을 로드하고 실행합니다.
7. 사용자 인터페이스 실행
   - 사용자 인터페이스(데스크톱 환경, 로그인 화면 등)가 실행되어 사용자가 컴퓨터를 사용할 수 있게 됩니다



컴퓨터 구조(Booting, CPU 작동원리, POST)

바이오스 단계

- PC의 전원 스위치를 켜면 제일 먼저 바이오스BIOS, Basic Input/Output System가 동작
- 바이오스는 보통 ROM에 저장되어 있어 흔히 ROM-BIOS라고 부름
- 바이오스는 PC에 장착된 기본적인 하드웨어(키보드, 디스크 등)의 상태를 확인한 후 부팅 장치를 선택하여 부팅 디스크의 첫 섹터에서 512B를 로딩함
- 512B를 마스터 부트 레코드MBR라고 하며, 여기에는 디스크의 어느 파티션에 2차 부팅 프로그램(부트 로더)이 있는지에 대한 정보가 저장되어 있음. MBR은 부트 로더를 찾아 메모리에 로딩하는 작업까지 수행

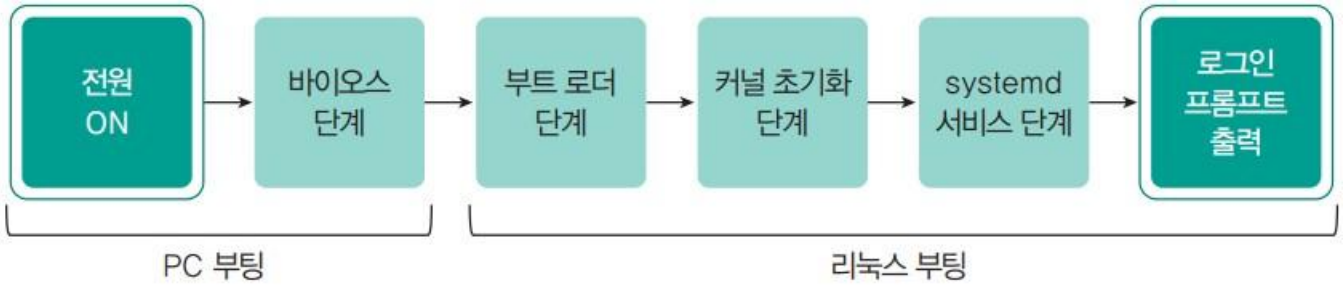
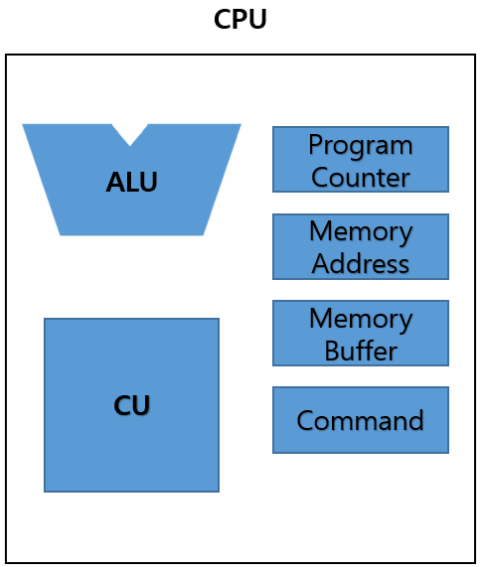


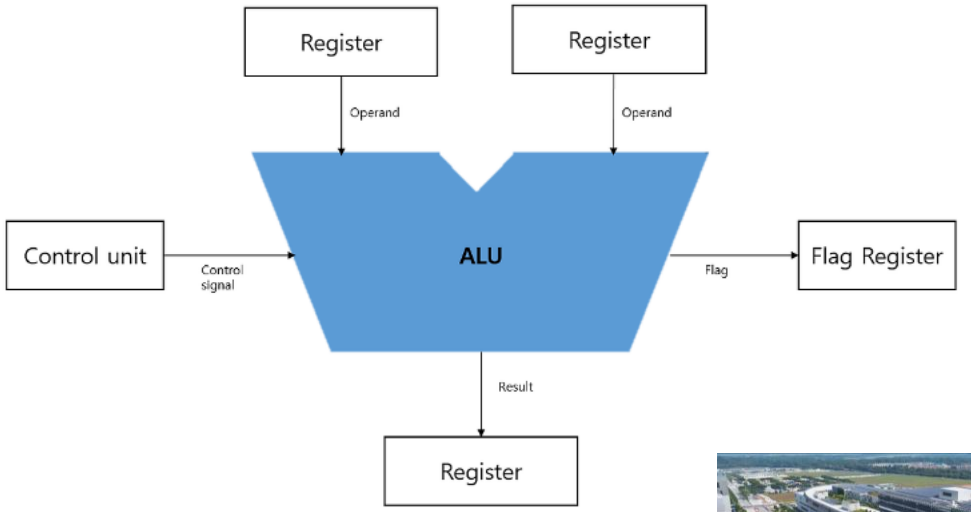
그림 7-1 리눅스의 부팅 과정

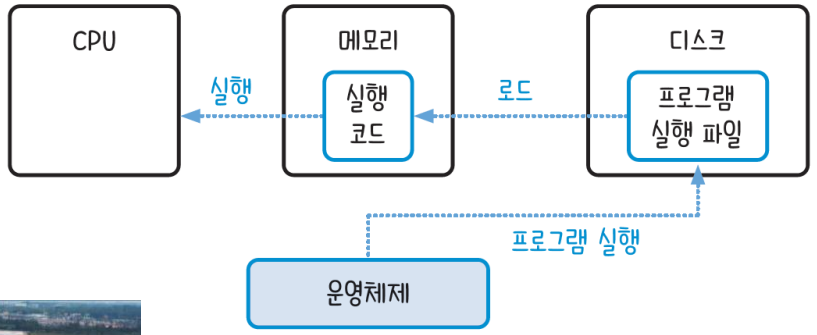
컴퓨터 구조(Booting, CPU 작동원리, POST)

CPU란 무엇인가?

컴퓨터의 뇌라고 할 수 있는 중앙처리장치(Central Processing Unit, CPU)는 주 기억장치인 메모리에서 명령어를 읽어 들이고 이를 해석하여 수행하는 작업을 맡는다. CPU의 주요 구성 요소로는 산술 및 논리 연산을 수행하는 ALU(Arithmetic Logic), 명령어의 순서와 수행을 제어하는 제어 유닛(Control Unit), 그리고 중간 결과와 작업 상태를 저장하는 레지스터(Register)가 있다.







Application 작동원리(마이크로 프로세서, 메모리, 저장장치)

마이크로 프로세서

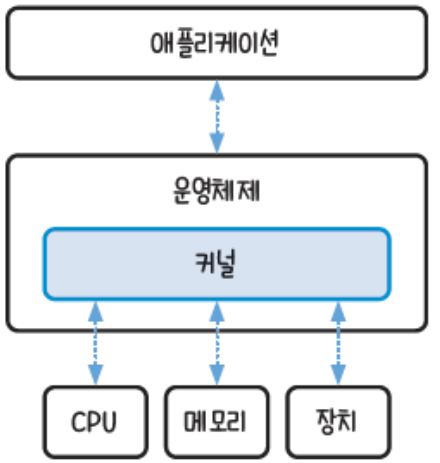
- 마이크로프로세서(Microprocessor)는 산술 논리 장치, 제어 장치, 레지스터를 하나의 단일체 집적회로로 구성한 것을 의미한다.
- CPU와의 차이 : CPU뿐만 아니라 GPU(Graphic Processing Unit), DSP(Digital Signal Processor)도 포함

커널

- 하드웨어를 초기화해 사용할 수 있게 하는 운영체제의 핵심 부분. 여러 소프트웨어가 운영체제에서 잘 작동할 수 있도록 메모리와 프로세스를 관리하고, 네트워크를 연결하는 등 주요 기능을 제공

파일, 파일 시스템

- 파일 : 정보를 저장하는 기본 단위
- 파일시스템 : HDD, SSD 등 저장 장치에 파일을 저장하고 관리하는 소프트웨어. 파일 시스템은 파일의 이름, 크기, 저장 위치를 저장하고, 파일에 대한 읽기/쓰기/실행을 제어하며, 파일의 권한을 관리하고 저장 장치에 대한 접근도 제어



네트워크 시스템

- 네트워크 드라이버(network driver) : 유선랜, 와이파이, 블루투스 등 네트워크 장치를 초기화해
- 사용할 수 있게 하는 장치 드라이버
- 네트워크 스택(network stack) : 네트워크 장치를 통해 들어온 네트워크 트래픽을 처리하는 네트워크 프로토콜을
- 구현한 소프트웨어
- 네트워크 프로토콜(network protocol) : 네트워크에서 데이터를 주고받는 데 적용되는 통신 규약

Application 작동원리(마이크로 프로세서, 메모리, 저장장치)

장치

- 그래픽 카드, 랜 카드, 디스크 드라이브, 마우스, 키보드 등 컴퓨터에 연결해 사용하는 모든 기기

컴퓨터 주기억장치 메모리(RAM)

- 컴퓨터가 켜지는 순간부터 CPU 연산과 동작에 필요한 모든 내용이 저장되는 곳

컴퓨터 저장장치 메모리 (플래시메모리 Flash memory)

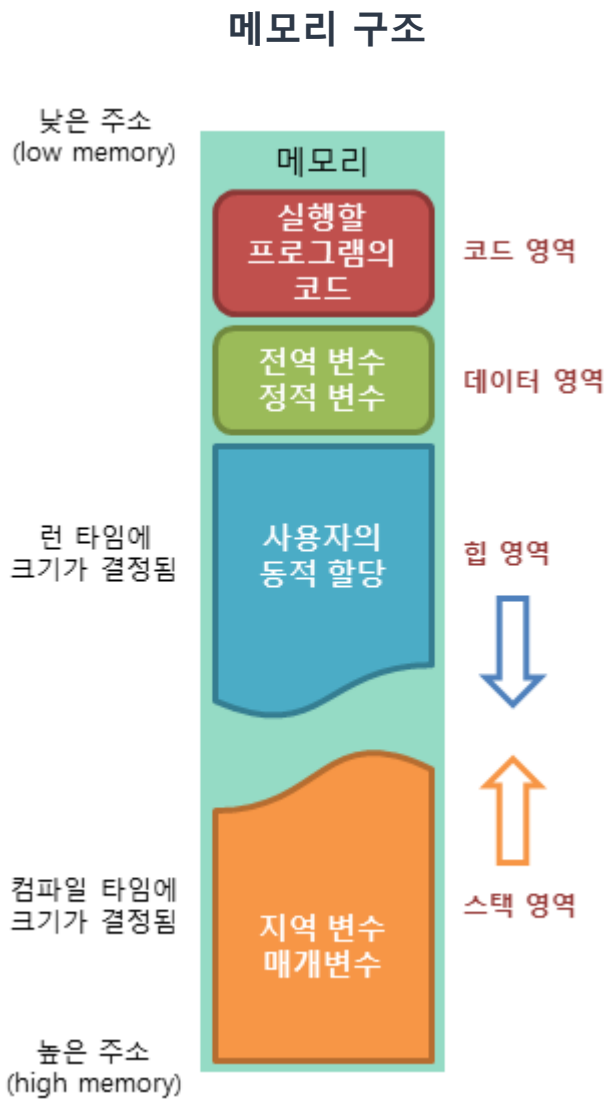
- RAM과 차별화된 비휘발성메모리 ROM중 여러 번 다시 작성할 수 있는 EPROM(Erasable PROM)의 한 종류인 플래시 메모리가 많이 사용

Application 작동원리(마이크로 프로세서, 메모리, 저장장치)

구분		종류		특징
주기억장치	RAM (휘발성)	SDRAM		1996-1999 기존 DRAM 파생형 동기식 전송방식
		DDR SDRAM		2000-2020 DDR / DDR2 / DDR3 / DDR4 / DDR5 시스템 메모리 Double Data Rate Synchronous Dynamic Random Access Memory 서버 / 데스크탑 / 노트북 메모리 크기와 핀수가 다름
		GDDR SDRAM		2000-2018 GDDR - GDDR6 SDRAM 그래픽카드 전용 메모리
		LPDDR SDRAM		2009-2020 LPDDR - LPDDR5 SDRAM (Low Power DDR) 모바일 플랫폼 작은크기 저발열 저전력
보조기억장치	USB 드라이브 셀저장방식 분류	NAND	SLC	Single Level Multicell 고급형 전문가형 / 저장속도 신뢰성 내구성 높음
			MLC	Multi Level Multicell 보급형 / 적절한 용량, 속도, 내구성
			TLC	Tripple Level Multicell 저가형 / 대용량
			QLC	Quad Level Multicell 초저가형 / 일회용 및 보급 판촉용
	SSD 인터페이스 분류	물리 구분	SATA	SerialATA / SATA Express 초기 하드디스크와 동일한 인터페이스
			mSATA	M.2 이전 노트북 소형화 기여
			M.2	AHCI와 NVMe 둘다 지원 / 슬림노트북 등 높은 성능과 속도
			기타	U.2 / PCI-Express / SAS / USB / 메모리카드(Flashcard)
		논리 구분	NVMe	Non-Volatile Memory (PCI-E x4규격)
			AHCI	고급 호스트 컨트롤러 인터페이스 Advanced Host Controller Interface (SATA규격)
			UASP	USB 3.0 기준 빠른 대역폭

플래시 카드 Flash Card	CF Compact Flash	1994년 샌디스크 규격 / 디지털카메라 내비게이션 사용
	SD Secure Digital	1999년 샌디스크 파나소닉 도시바 공동개발 파생상품 miniSD / microSD / 어댑터 사용가능
	메모리스틱	1998년 소니 독자규격 / 소니 알파 캠코더 핸디캠 노트북 바이오 소니 워크맨에 사용
	xD 픽처카드	2002년 올림푸스와 후지필름이 만든 extreme Digital 카드
	eMMC	embeded Multi Media Card 플래시메모리에 컨트롤러를 통합한 저가 저사양 태블릿 컴퓨터나 스마트폰 MP3에 많이 쓰임

Application 작동원리(마이크로 프로세서, 메모리, 저장장치)



프로그램이 실행되기 위해서는 먼저 프로그램이 메모리에 로드(load)되어야 합니다.
또한, 프로그램에서 사용되는 변수들을 저장할 메모리도 필요합니다.
따라서 컴퓨터의 운영체제는 프로그램의 실행을 위해 다양한 메모리 공간을 제공하고 있습니다.
프로그램이 운영체제로부터 할당받는 대표적인 메모리 공간은 4종류입니다.

- **코드(code) 영역**
- **데이터(data) 영역**
- **스택(stack) 영역**
- **힙(heap) 영역**

1. 코드(code) 영역

메모리의 코드(code) 영역은 실행할 프로그램의 코드가 저장되는 영역으로 텍스트(code) 영역이라고도 부릅니다.

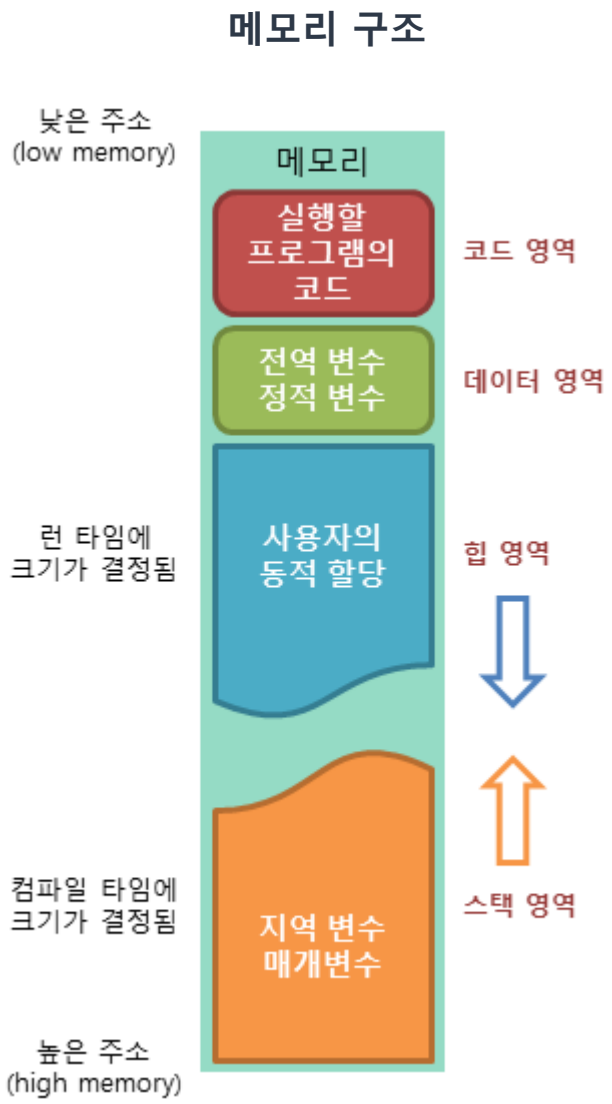
CPU는 코드 영역에 저장된 명령어를 하나씩 가져가서 처리하게 됩니다.

2. 데이터(data) 영역

메모리의 데이터(data) 영역은 프로그램의 전역 변수와 정적(static) 변수가 저장되는 영역입니다.

데이터 영역은 프로그램의 시작과 함께 할당되며, 프로그램이 종료되면 소멸합니다.

Application 작동원리(마이크로 프로세서, 메모리, 저장장치)



스택(Stack) 영역

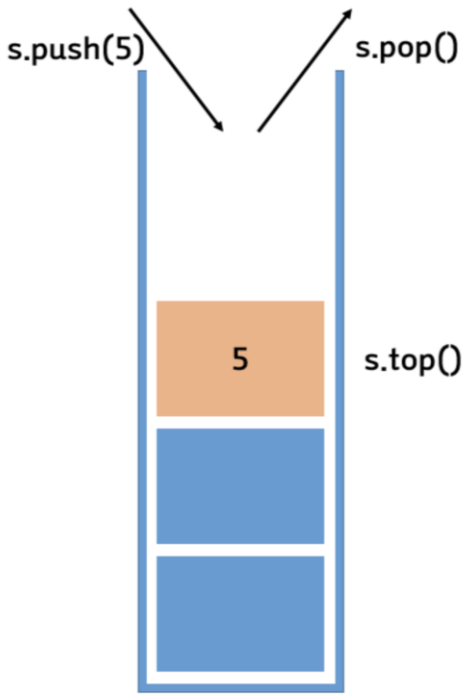
메모리의 스택(stack) 영역은 함수의 호출과 관계되는 지역 변수와 매개변수가 저장되는 영역입니다.
스택 영역은 함수의 호출과 함께 할당되며, 함수의 호출이 완료되면 소멸합니다.
이렇게 스택 영역에 저장되는 함수의 호출 정보를 스택 프레임(stack frame)이라고 합니다.
스택 영역은 푸시(push) 동작으로 데이터를 저장하고, 팝(pop) 동작으로 데이터를 인출합니다.
이러한 스택은 후입선출(LIFO, Last-In First-Out) 방식에 따라 동작하므로, 가장 늦게 저장된 데이터가 가장 먼저 인출됩니다.
스택 영역은 메모리의 높은 주소에서 낮은 주소의 방향으로 할당됩니다.

힙(Heap) 영역

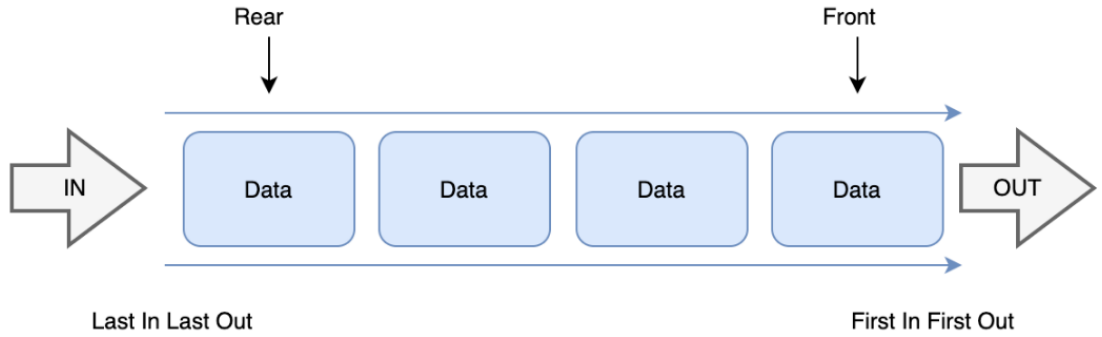
메모리의 힙(heap) 영역은 사용자가 직접 관리할 수 있는 '그리고 해야만 하는' 메모리 영역입니다.
힙 영역은 사용자에 의해 메모리 공간이 동적으로 할당되고 해제됩니다.
힙 영역은 메모리의 낮은 주소에서 높은 주소의 방향으로 할당됩니다.

스택	힙
• 매우 빠른 액세스 • 변수를 명시적으로 할당 해제 할 필요가 없습니다. • 공간은 CPU에 의해 효율적으로 관리됩니다. • 지역 변수 • 스택 크기 제한 (OS에 따라 다름) • 변수의 크기를 조정할 수 없습니다.	• 변수는 전역적으로 액세스 할 수 있습니다. • 메모리 크기 제한 없음 • (상대적으로) 느린 액세스 • 효율적인 공간 사용을 보장하지 못하면 메모리 블록이 할당 된 후 시간이 지남에 따라 메모리가 조각화되어 해제 될 수 있습니다. • 메모리를 관리해야합니다 (변수를 할당하고 해제하는 책임이 있습니다)

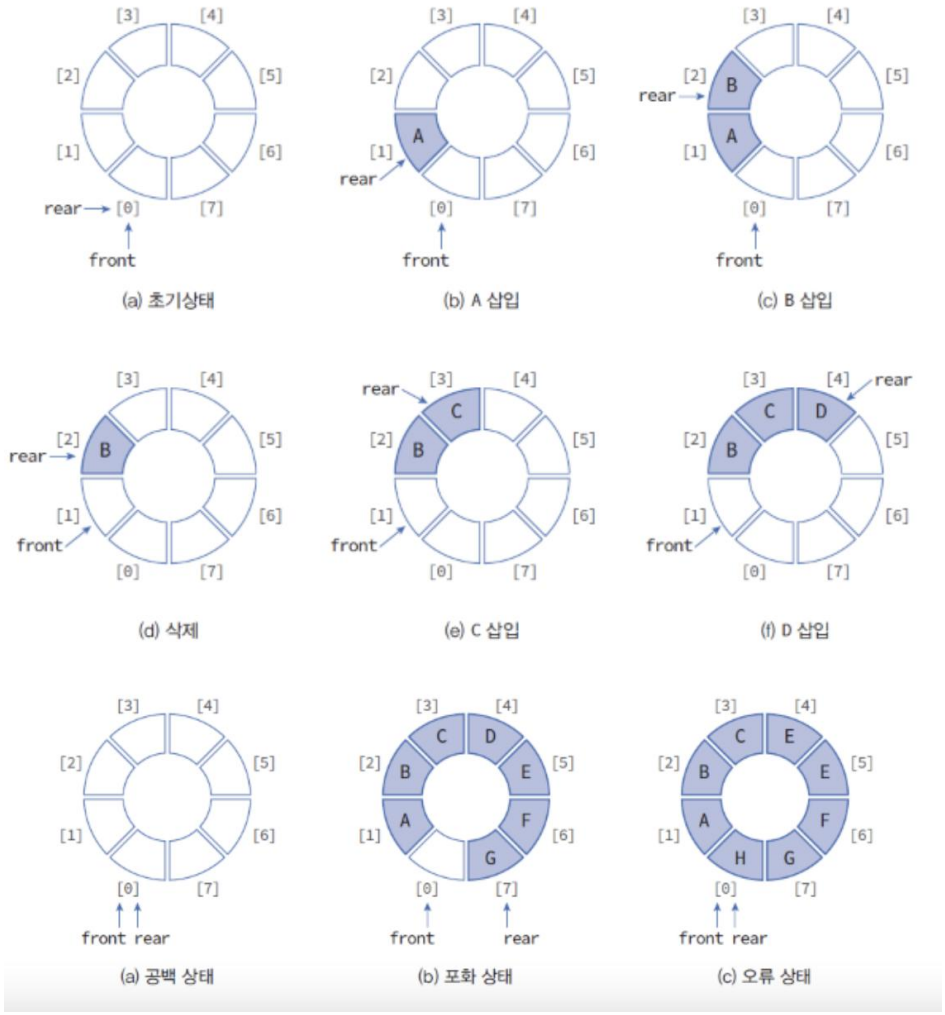
Stack, Queue, Circular Queue, Deque



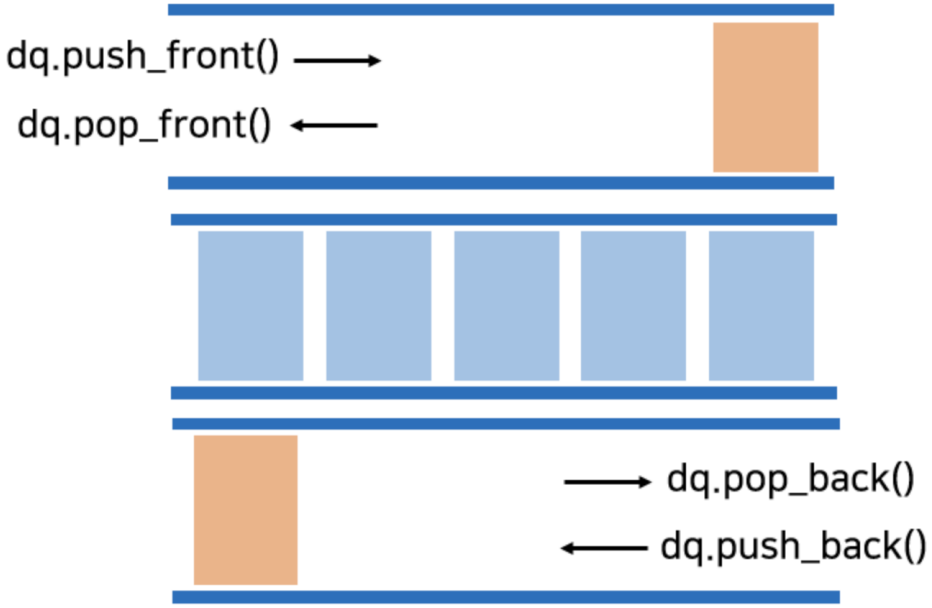
Stack



Queue



Circular Queue



Deque
(Double Ended Queue)

컴퓨터 구조(Booting, CPU 작동원리, POST)

프로그램이란

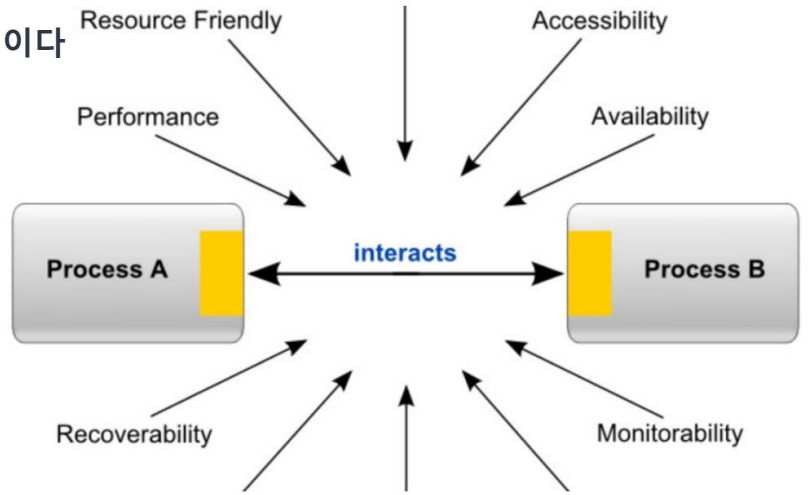
프로그램의 의미는 어떤 작업을 하기 위해 해야할 일들을 순서대로 나열한 것으로 쉽게 말해 컴퓨터에서 어떤 작업을 위해 실행할 수 있는 '정적인 상태'의 파일이라고 볼 수 있다.
컴퓨터에서의 '프로그램'은 사용자가 원하는 일을 처리할 수 있도록 프로그래밍 언어를 사용하여 올바른 수행절차를 표현해 놓은 명령어들의 집합이다.
그에 필요한 데이터를 묶어 놓은 파일로 보조 기억장치에 저장되어있다.

프로세스(Process)란 무엇인가

프로그램이 실행되서 돌아가고 있는 상태, 컴퓨터에서 연속적으로 실행되고 있는 '동적인 상태'의 컴퓨터 프로그램이다

[특징]

- 프로세스는 각각 Code, Data, Stack, Heap의 구조로 되어있는 독립된 메모리 영역을 할당 받는다.
- 다른 프로세스의 자원에 접근하려면 프로세스간의 통신(IPC)을 사용해야한다.
- 프로세스는 최소 하나 이상의 스레드를 포함한다.
- 각 프로세스는 별도의 주소 공간에서 실행되며, 서로 독자적인 메모리 공간을 갖기 때문에 서로 메모리 공간을 공유할 수 없다. 즉, 다른 프로세스의 변수나 자료구조에 접근할 수 없다.



스레드(Thread)란 무엇인가

- 스레드(Thread)는 프로세스가 할당 받은 자원을 이용하는 실행 단위이자, 프로세스의 특정한 수행 경로이자 프로세스 내에서 실행되는 여러 흐름의 단위이다.
- 스레드는 프로세스 내에서 프로세스의 자원을 이용해서 실제로 작업을 수행하는 일꾼이다.
- 스레드가 소속된 프로세스가 운영체제로부터 자원을 할당받으면 그 자원을 스레드가 사용한다.
- 프로세스는 최소 한 개 이상의 스레드를 가지며 이 스레드를 메인 스레드(main thread)라고 한다.

[스레드의 특징]

- 각 스레드는 독자적인 스택(Stack) 메모리를 갖는다.
- 스레드는 프로세스 내에서 각각 스택만 할당받고 Code, Data, Heap 영역은 공유한다.
- 스레드는 한 프로세스 내에서 동작되는 여러 실행의 흐름으로, 프로세스 내의 주소공간이나 자원들을 같은 프로세스 내의 스레드끼리 공유하며 실행된다.
- 각각의 스레드는 별도의 레지스터와 스택을 갖고 있지만, 힙 메모리는 서로 읽고 쓸 수 있다.
- 한 스레드가 프로세스 자원을 변경하면, 다른 이웃 스레드(sibling thread)도 그 변경 결과를 즉시 볼 수 있다.
- 스레드는 메모리를 공유하기 때문에 동기화, 데드락 등의 문제가 발생 할 수 있다.
- 스레드는 대부분의 현대 운영체제가 지원하고 있으며, 이와 관련된 주요 라이브러리로는 POSIX Pthreads, Windows threads, Java threads 가 있다.

프로세스(Process)와 스레드(Thread)

- 프로세스와 스레드의 관계 : 프로세스는 스레드의 컨테이너이다. 스레드의 정보를 담고있는 것에 불과하다.
- 프로세스와 스레드의 차이점 : 프로세스는 각 작업(Task)마다 운영체제로부터 자원을 할당받기 위해 시스템 콜을 하는 부담이 생기지만 멀티 스레드를 사용한다면 시스템 콜을 한번만 해도 되기 때문에 효율적이다.
- 또한 IPC 방식보다는 스레드 간 통신이 덜 복잡하고 시스템 자원 사용이 더 적으므로 통신의 부담도 줄일 수 있다.

멀티프로세스와 멀티스레드

멀티 스레딩이란

- 하나의 응용프로그램을 여러 개의 스레드로 구성하고 각 스레드로 하여금 하나의 작업을 처리하도록 하는 것이다.
- 윈도우, 리눅스 등 많은 운영체제들이 멀티 프로세싱을 지원하고 있지만 멀티 스레딩을 기본으로 하고 있다.
- 웹 서버는 대표적인 멀티 스레드 응용 프로그램이다.

멀티 스레딩의 장점

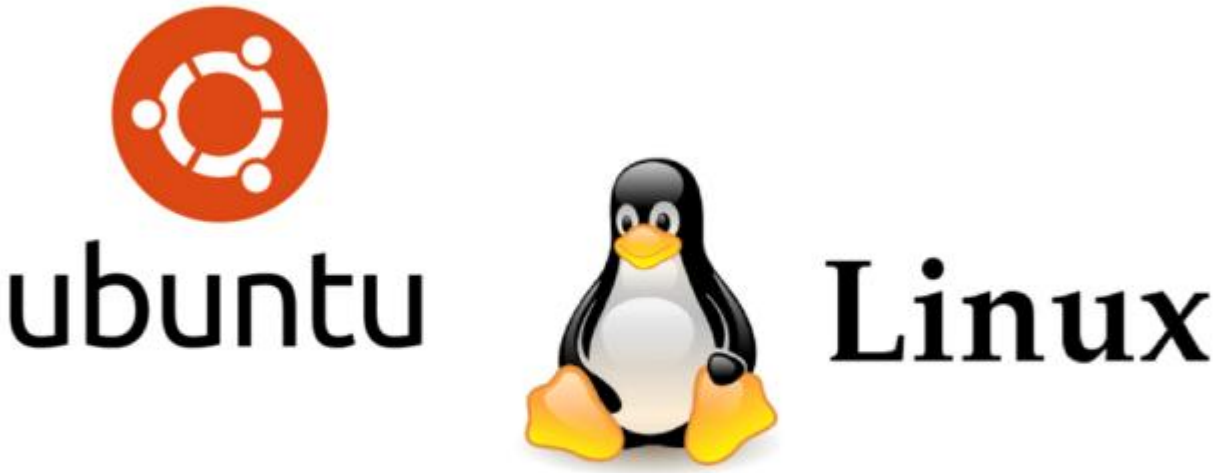
- 시스템 자원 소모 감소 (자원의 효율성 증대)
- 프로세스를 생성하여 자원을 할당하는 시스템 콜이 줄어들어 자원을 효율적으로 관리할 수 있다.
- 시스템 처리량 증가 (처리 비용 감소)
- 스레드 간 데이터를 주고 받는 것이 간단해지고 시스템 자원 소모가 줄어들게 된다.
- 스레드 사이의 작업량이 작아 Context Switching이 빠르다.
- 간단한 통신 방법으로 인한 프로그램 응답 시간 단축
- 스레드는 프로세스 내의 Stack 영역을 제외한 모든 메모리를 공유하기 때문에 통신의 부담이 적다.

멀티 스레딩의 단점

- 주의 깊은 설계가 필요하다.
- 디버깅이 까다롭다.
- 단일 프로세스 시스템의 경우 효과를 기대하기 어렵다.
- 다른 프로세스에서 스레드를 제어할 수 없다. (즉, 프로세스 밖에서 스레드 각각을 제어할 수 없다.)
- 멀티 스레드의 경우 자원 공유의 문제가 발생한다. (동기화 문제)
- 하나의 스레드에 문제가 발생하면 전체 프로세스가 영향을 받는다.

*멀티프로세스 대신 멀티 스레드를 사용하는 이유는?
프로그램을 여러 개 키는 것보다 하나의 프로그램 안에서 여러 작업을 해결하는 것이 더 효율적이기 때문이다.

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)



리눅스와 운영체계

운영체제란 무엇인가?

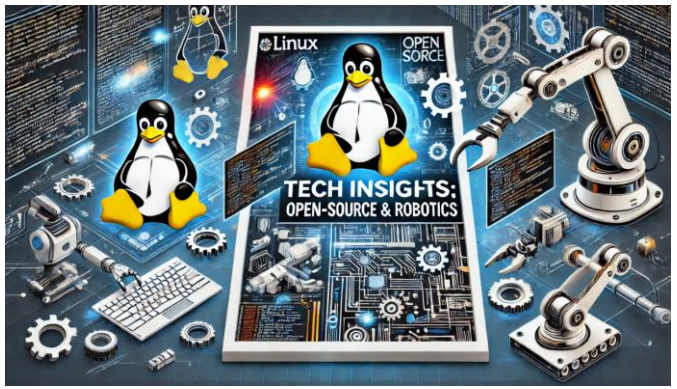
운영체제(Operating System, OS)는 사용자가 컴퓨터를 사용하기 위해 필요한 소프트웨어
일반적으로 컴퓨터를 사용하면서 실행한 모든 프로그램들은 운영체제에서 관리하고 제어한다

운영체제 목적

컴퓨터의 하드웨어 관리 + 사용자 편의 제공
컴퓨터의 성능을 높이고(performance), 사용자에게 편의성 제공(Convenience)을 목적으로 하는 컴퓨터 하드웨어 관리하는 프로그램

리눅스란

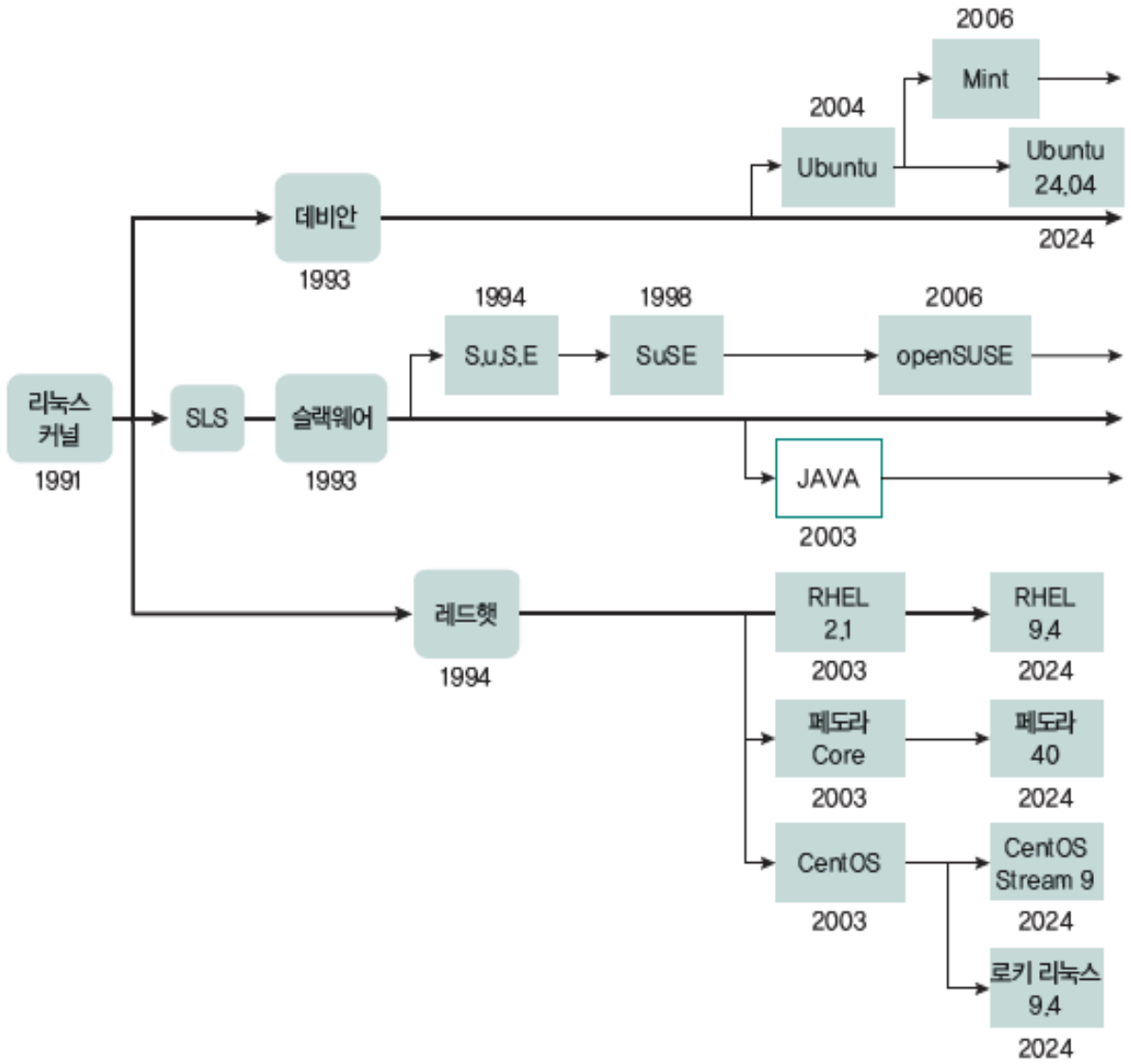
리눅스(Linux)는 리누스 토르발즈가 유닉스(Unix)에 기반하여 만든 운영체제



특징 :

- 독립된 플랫폼을 갖는 운영체제
- 빠른 업데이트
- 강력한 네트워크 지원
- 다중 직업과 가상 터미널 환경지원
- 유닉스 호환
- 공개형 오픈 소스 운영체제
- 다중 사용자 환경 지원
- 저사양 컴퓨터에서 서버 구축 가능
- 이식성과 확장성

리눅스와 운영체계


리눅스 커널
1991
SLS
슬랙웨어
1993
데비안
1993
레드햇
1994
S.u.S.E
1994
SuSE
1998
openSUSE
2006
JAVA
2003
Ubuntu
2004
Mint
2006
Ubuntu 24.04
2024
RHEL 2.1
2003
RHEL 9.4
2024
페도라 Core
2003
페도라 40
2024
CentOS
2003
CentOS Stream 9
2024
로키 리눅스 9.4
2024

운영체제의 역사

연대	주요 이슈
1940~1950년대	처음 등장한 컴퓨터인 에니악(ENIAC)에는 운영체제라고 할 만한 것이 없었으며 사용자가 직접 하드웨어를 제어하였다.
1960년대 초기	본격적인 운영체제라고 할 수 있는 일괄처리시스템이 등장하였다. 일괄처리시스템은 여러 응용 프로그램을 등록된 순서에 따라 순차적으로 실행하는 시스템이다.
1960년대 후반	시분할시스템(Time Sharing System)이 등장하였다. 시분할시스템은 CPU 시간을 작은 단위로 나누고 이를 여러 프로그램에 돌아가며 할당하여 실행하는 시스템으로, 사용자 입장에서는 동시에 여러 프로그램이 실행되는 것처럼 보인다.
1970년대	현대 운영체제의 기본 기술이 들어간 최초의 운영체제인 유닉스가 C 언어로 구현되어 빠르게 퍼져 나갔다.
1980년대	개인용 컴퓨터가 보급되고, MS-DOS가 등장하였다. 이 시기에 GUI 환경이 제공되기 시작하였다.
1990년대	네트워크 기술이 발전하면서 웹이 대중화되기 시작하였다. 리눅스가 등장하였으며 오픈소스 운동이 활성화되었다.
2000년대 이후	안드로이드, iOS 등 모바일 운영체제가 등장하였다. 또한 가상머신과 대용량 병렬처리 기술이 등장하였다.

리눅스와 운영체계

리눅스와 유닉스

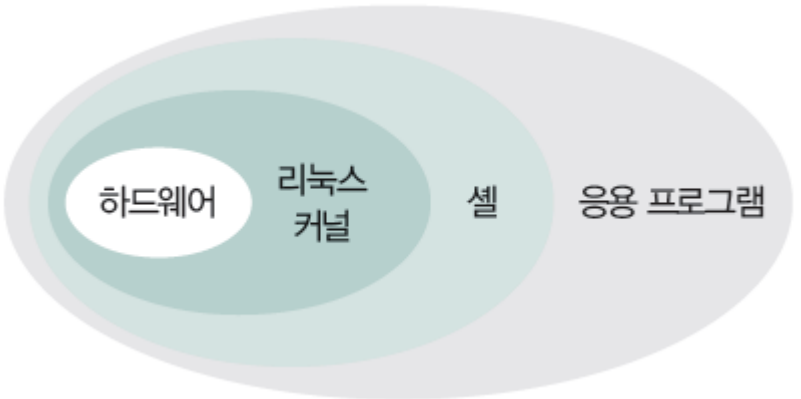
- 리눅스: 유닉스 계열 운영체제, '리누스가 만든 유닉스'라는 의미
- 유닉스: 1969년 AT&T 벨연구소에서 개발, 1971년 C 언어로 재개발된 최초의 고급 프로그래밍 언어 기반 운영체제. 상용화 버전(시스템 계열)과 오픈소스 버전(BSD 계열)으로 발전
- 리눅스의 등장: 1980년대 후반, BSD 유닉스가 AT&T와 법적 공방 중 리누스 토르발스가 개발

리눅스의 시작과 발전

- 핀란드 헬싱키대학교 학생인 리누스 베네딕트 토르발스가 교육용 운영체제 미닉스를 참고해 개발
- 오픈소스 운동에 합류되며 발전, 현재는 서버, 슈퍼컴퓨터, 임베디드 시스템, 모바일 기기에서 사용, 안드로이드 운영체제도 리눅스 기반
- 리눅스 커널: 1991년 8월 처음 공개, 현재 최신 버전 6.9(2024년 5월 25일 기준)

리눅스의 특징

- 리눅스는 공개 소프트웨어이며 무료로 사용할 수 있다.
- 유닉스와 완벽한 호환성을 유지한다.
- 서버용 운영체제로 많이 사용된다.
- 편리한 GUI 환경을 제공한다.

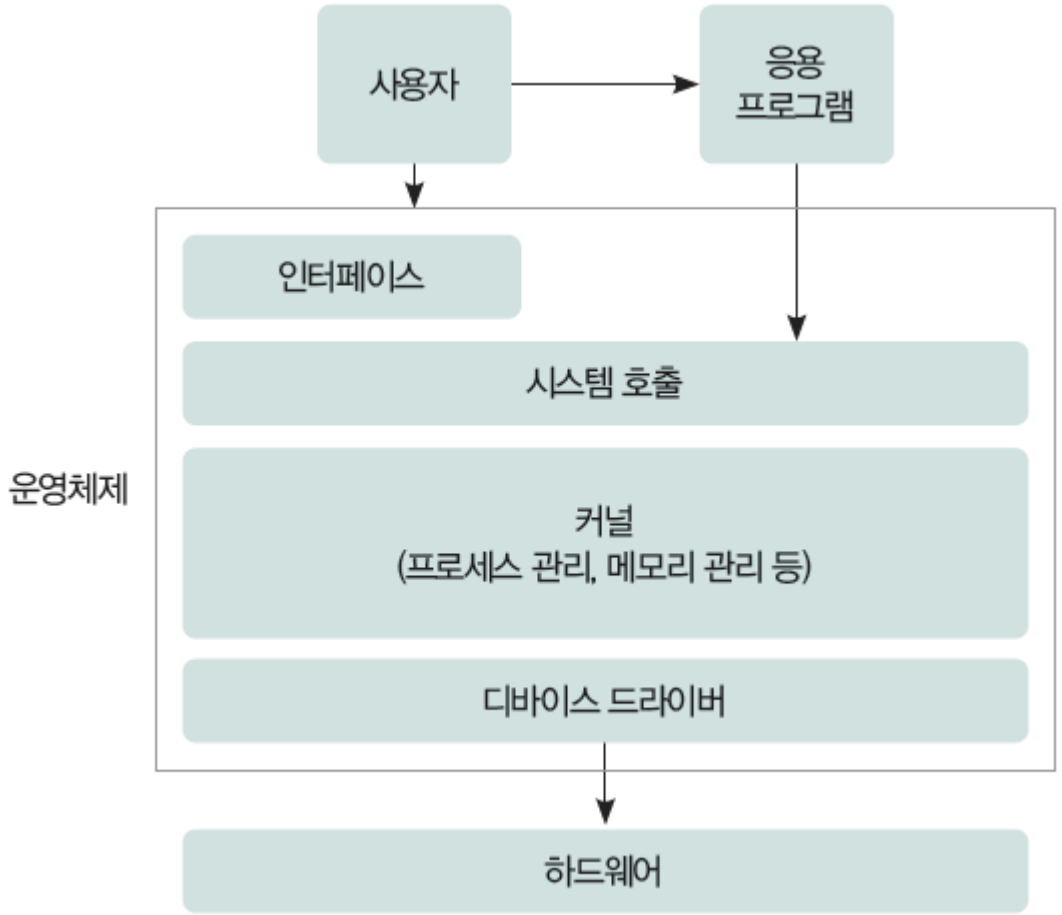


리눅스와 운영체계

- 운영체제의 구성요소
  - 커널 : 시스템의 핵심 기능을 담당
  - 파일 시스템 : 데이터 저장 및 관리
  - 디바이스 드라이버 : 하드웨어와의 통신
  - 시스템 호출 : 운영체제 서비스 접근
  - 인터페이스 : 사용자와의 상호작용
- CPU의 동작 모드
  - 커널 모드: 하드웨어 자원에 직접 접근 가능
  - 사용자 모드: 시스템 호출을 통해 자원 접근

운영체제의 기능

- 프로세스 관리: 프로세스 생성, 스케줄링, 관리
- 메모리 관리: 메모리 할당/해제, 가상 메모리
- 파일 시스템 관리: 파일/디렉터리 생성, 읽기, 쓰기, 삭제
- 입출력 관리: 데이터 입출력, 버퍼링, 스케줄링
- 자원 관리 및 보호: 하드웨어 자원 관리, 충돌 방지
- 사용자 인터페이스 제공: GUI 및 명령행 인터페이스 제공

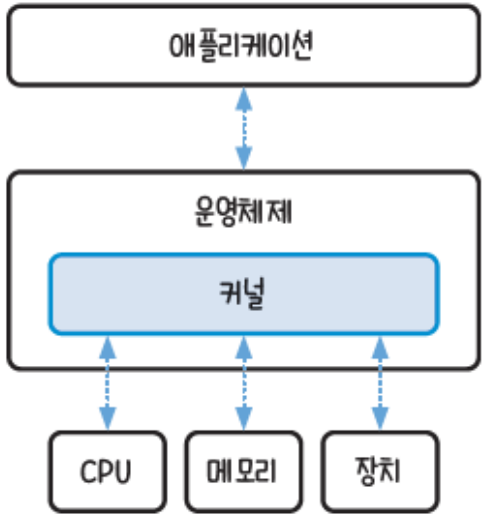


커널

- 하드웨어를 초기화해 사용할 수 있게 하는 운영체제의 핵심 부분. 여러 소프트웨어가 운영체제에서 잘 작동할 수 있도록 메모리와 프로세스를 관리하고, 네트워크를 연결하는 등 주요 기능을 제공

파일, 파일 시스템

- 파일 : 정보를 저장하는 기본 단위
- 파일시스템 : HDD, SSD 등 저장 장치에 파일을 저장하고 관리하는 소프트웨어. 파일 시스템은 파일의 이름, 크기, 저장 위치를 저장하고, 파일에 대한 읽기/쓰기/실행을 제어하며, 파일의 권한을 관리하고 저장 장치에 대한 접근도 제어



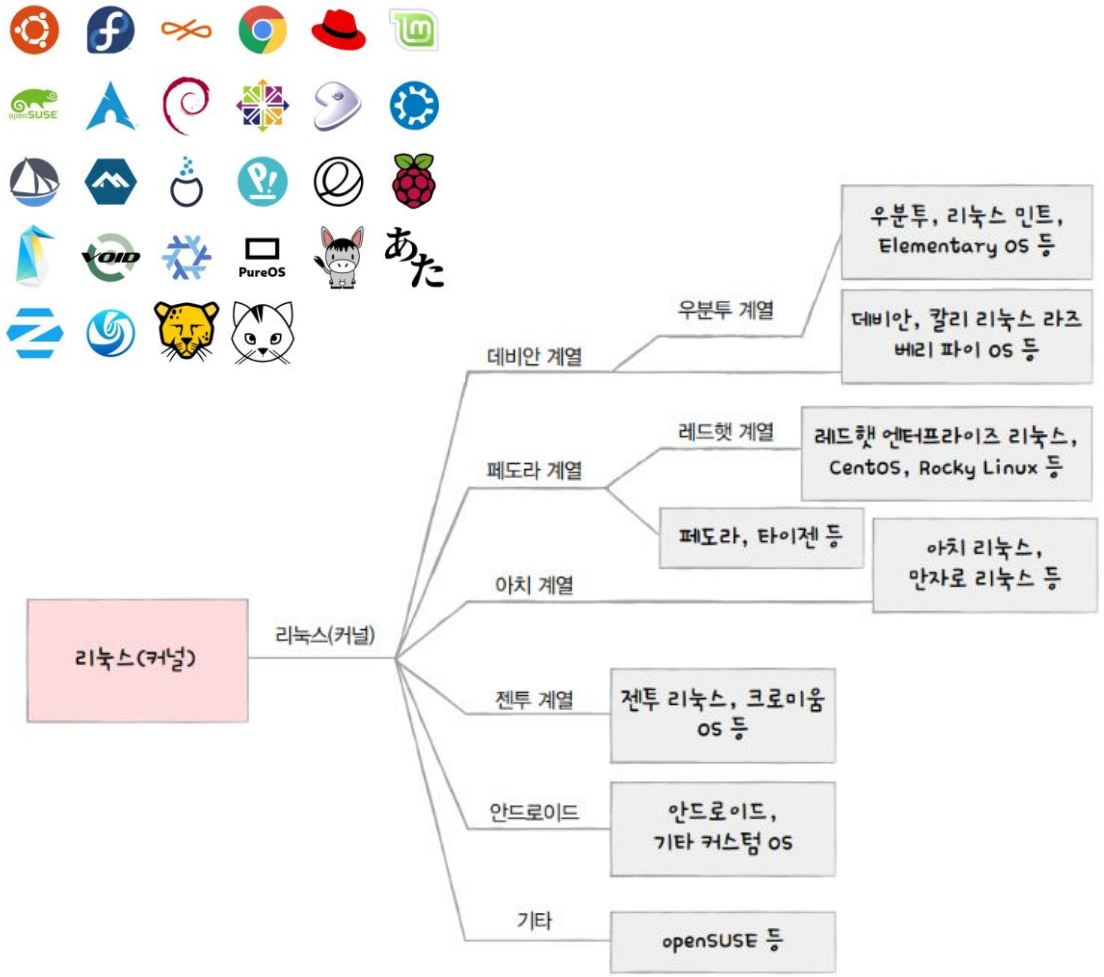
네트워크 시스템

- 네트워크 드라이버(network driver) : 유선랜, 와이파이, 블루투스 등 네트워크 장치를 초기화해 사용할 수 있게 하는 장치 드라이버
- 네트워크 스택(network stack) : 네트워크 장치를 통해 들어온 네트워크 트래픽을 처리하는 네트워크 프로토콜을 구현한 소프트웨어
- 네트워크 프로토콜(network protocol) : 네트워크에서 데이터를 주고받는 데 적용되는 통신 규약

리눅스와 운영체계

리눅스의 계열

공개형 오픈 소스 운영체제. 마이크로 소프트 - 윈도우, 애플 – IOS 와 달리 리눅스는 종류가 많다



레드햇 계열

페도라에서 파생된 배포판들이다. 패키지 형식은 .rpm이며 패키지 관리자로 yum을 사용하는 것이 특징이다. 서버용으로 사용되는 경우가 대부분이다. 얼마 없는 상용 리눅스 중에서도 굉장히 잘나가는 편인데, 이는 서버 시장이 주 타겟인 배포판이기 때문

맨드리바/마제야 계열

마제야에서 파생된 배포판들. 본래 맨드리바 기반이었으나, 맨드리바가 개발 중단되면서 독립하였다. 쉬운 사용성을 추구하며, KDE를 주력 데스크톱 환경으로 밀고 있는 몇 안되는 배포판들이다.

데비안 계열

데비안에서 파생된 배포판들. 패키지 형식은 .deb이며, 패키지 관리자로 apt를 이용한다.

그외 우분투, 아치, 슬랙웨어 등등

다양한 리눅스 종류

공개형 오픈 소스 운영체제. 마이크로 소프트 - 윈도우, 애플 – IOS 와 달리 리눅스는 종류가 많다

▶최초의 리눅스 배포판 :SLS

1992년 5월 피터 맥도날드에 의해 만들어진 소프트랜딩 리눅스 시스템(Softlanding Linux System: SLS)이 리눅스 최초의 배포판이다. 출시 당시에는 가장 인기 있는 리눅스 배포판이었지만 사용자들에 의해 버그가 다소 존재하는 것으로 파악되었고 이것은 곧 다른 리눅스 배포판의 등장을 알리게 되는 계기가 되었다. 패트릭 볼커딩은 SLS에 존재하는 버그를 잡기 시작하는데 이 결과로 만들어진 리눅스 배포판이 슬렉웨어이다.

▶슬랙웨어 (Slackware): 현재까지 살아있는 **가장 오래된 배포판**

슬랙웨어는 현재까지 살아있는 가장 오래된 배포판이예요.

하지만 뭔가 프로그램을 설치할 때 필요한걸 알아서 따따따따따 설치해주는,, 의존성 문제를 **우분투나 레드햇처럼 자동으로** 해결해주지 않기 때문에,, 초보자가 사용하는 용은 아니죠 ㅎㅎ

현재 슬렉웨어를 많이 사용하는 것 같지는 않습니당.

▶수세 (OpenSUSE)

수세 리눅스는 독일에서 출시된 배포판으로 유럽에서 가장 인기를 누리고 있습니다. **오픈 수세는 상당히 안정적인 리눅스**입니다. 믿고쓰는 배포판이라는 말이 있을정도 ㅎㅎ, **다만 저장소가 우분투나 민트에 비해 많지 않다는 단점과 국내에 정보가 많지 않다는 단점**이 있습니다.

수세 리눅스(SUSE Linux) 배포판은 컴퓨터 운영체제이다. 리눅스 커널을 최상위로 하여 만들어졌으며 다양한 프로젝트로부터 나온 시스템 소프트웨어와 응용 소프트웨어를 포함하여 배포한다. 수세 리눅스는 본래 독일에서 나왔으며 주로 유럽에서 개발된다.

▶레드햇 (Red Hat)

레드햇은 **기업용 서버 OS**로 가장 인기가 있죠! 그래서 **실제 현업에서 가장 널리 사용되는 OS**이기도 합니다.

예전에는 레드햇 리눅스는 유로버전과 무료버전을 모두 배포하였으나 현재 **레드햇 리눅스의 의미는 상용으로 판매**되는 **레드햇 엔터프라이즈 리눅스(RHEL)만을 의미합니다.**

▶페도라 (Fedora) : 레드헷 계열

레드햇이 Red Hat 9.0을 마지막으로 더는 무료로 레드햇을 배포하지 않았는데, 그 대신 **페도라 프로젝트를 후원함으로써 일반인들도 계속 레드햇 리눅스의 연장선에 있는 페도라 리눅스를 사용할 수 있게 했습니다**. 무료인 페도라 리눅스는 상용인 RHEL에 포함될 새로운 기술을 미리 시험하는 용도로 사용됩니다. 즉 페도라 리눅스에서 안정화된 기능을 RHEL에 포함한것.

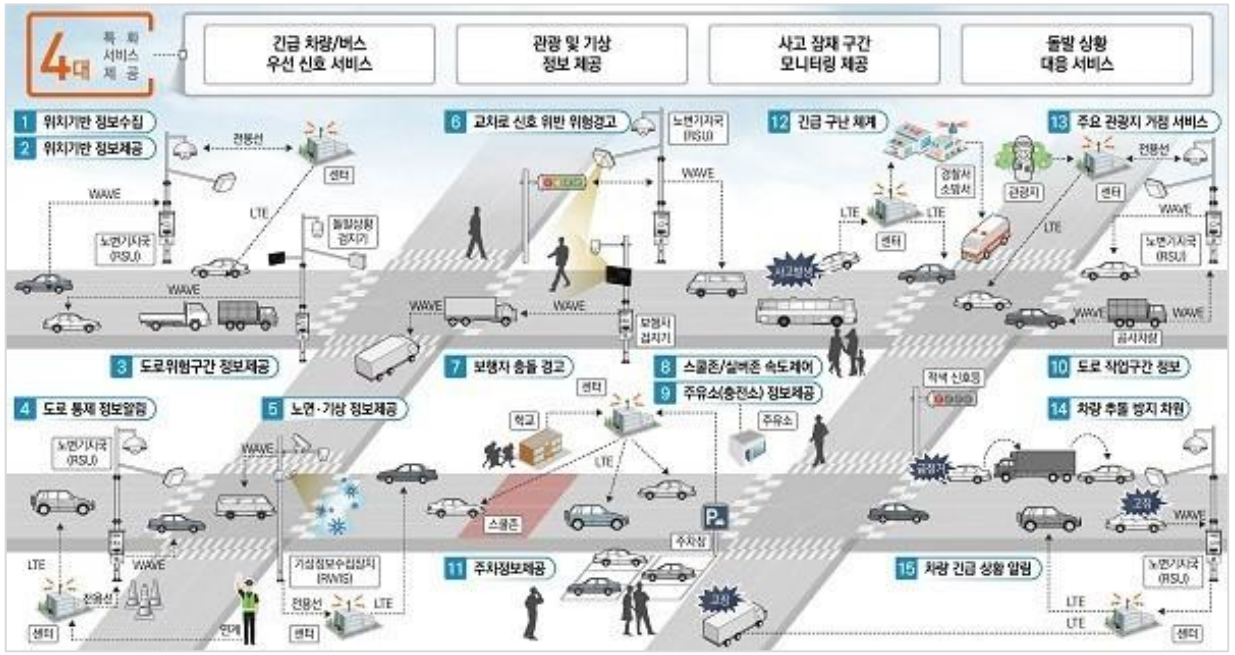
▶센트OS (centOS) : 레드헷 계열

RHEL이 유료로 변경하면서 반발에 의해 **무료버전으로 빌드한게 센트오에스**예요 ㅎㅎ, **센트OS는 레드햇 엔터프라이즈 기술을 그대로 사용할 수 있다는 장점**이 있습니다. 유료로 판매되는걸 어떻게 무료로 배포하지 할 수 있는데 GPL 라이선스 룰을 이용한 거예요. GPL은 소스를 받은 자가 그것을 재배포하는 것을 막지 않기 때문에 **RHEL을 구입한 자가 누구든간에 그는 레드햇에 소스를요청할 권**리가 있습니다. **이 점을 이용해서 상표권을 배제하고 배포된 리눅스가 centOS**입니다.

리눅스와 운영체계

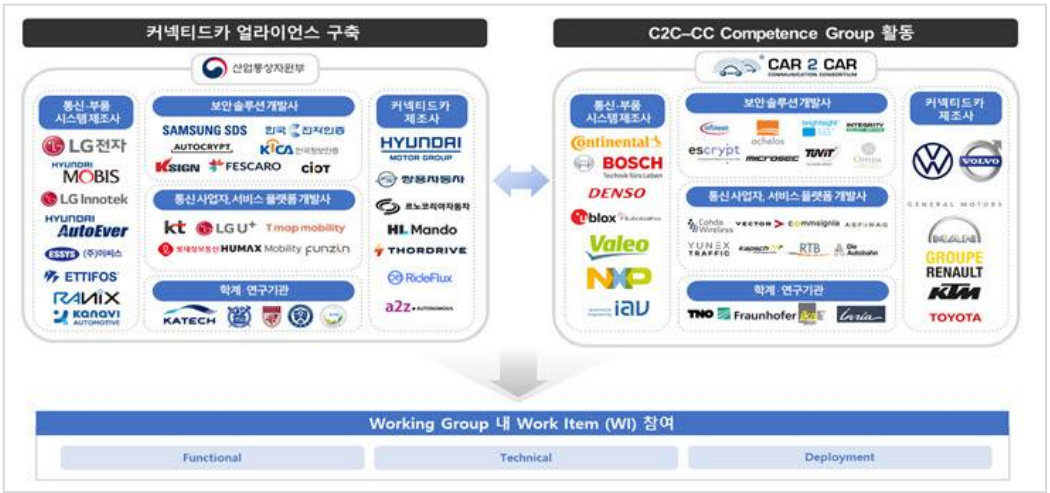
어디에 쓰일까

- 서버 : IT 서비스를 구성하기 위한 서버
- 클라우드 컴퓨팅 : 클라우드 서비스를 구축하기 위한 백엔드
- 임베디드 시스템 : PC나 서버 환경에 비해 제한적인 자원을 가진 경우
- 모바일 기기 : 스마트폰이나 태블릿 등



누가/왜 배워야 할까

- 컴퓨터와 관련한 직군
- 소프트웨어 개발자
- 시스템 관리자, 소프트웨어 엔지니어
- 네트워크 엔지니어
- 데이터 과학자나 AI 전문가



Smart Factory C-ITS

리눅스와 운영체계


KIOSK


1회용 발매기 · 교통카드 충전기
Ticket Vending and Card Reload Machine
KORAIL


P1


내 손안에 서울
104번 곧도착버스
104번 다음 도착버스
저상버스
버스내 혼잡도
104
곧도착
다음 3분
7021
곧도착
다음 6분
오늘오후 18.0℃
미세먼지 47
오후 4:49
400
8분
700
6분
707
9분
02-118
3분 이후 도착하는 버스안내
정류장 고유번호

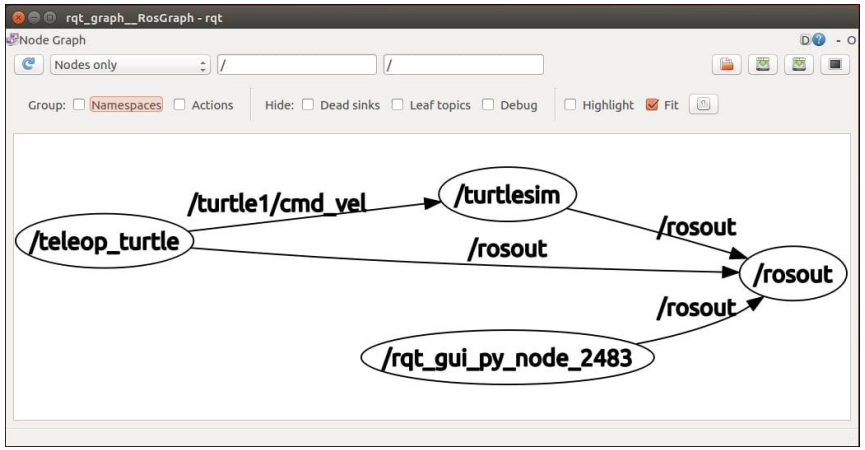

ifex 하이패스 승차권자동발매기 Tickets


영상약정 차량
차량 번호판 영상인식
후불 고지
통행료 납부
하이패스 차량
현행 하이패스 방식으로 통행료 자동납부

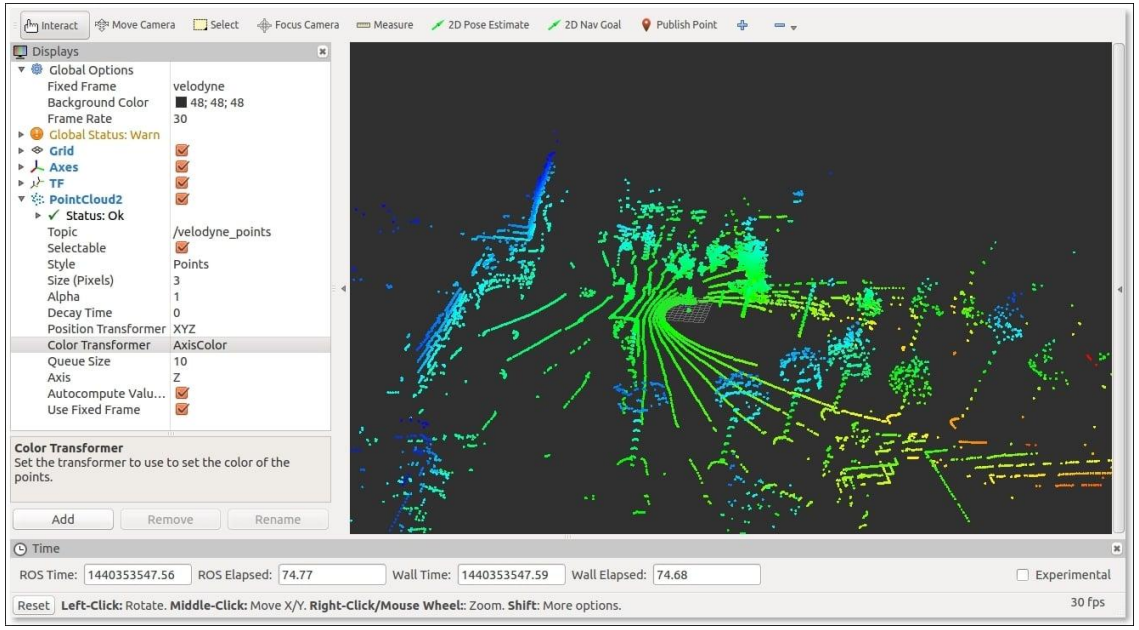
ROS와 리눅스

- 운영체계 라이선스 -> 오픈 API
- 개발 시간 단축 -> API, 생태계 활용
- 편리한 디버깅 도구 및 시각화
- 많은 사용자 및 생태계 -> 오픈 API

ROS는 다양한 생태계를 지향하여 특정 OS, 특정 프로그래밍 언어가 강제되지 않지만 해당 이유로 인해 본 강의에서는 ROS를 리눅스(우분투)에서 사용한다



시각화 도구

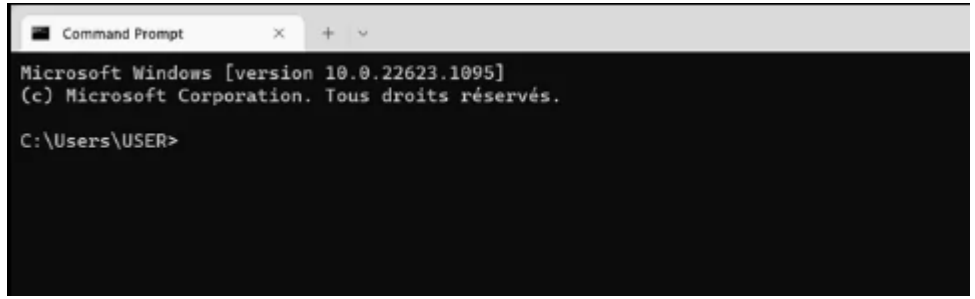


ROS에서 왜 파이썬을 쓰는가?

우리가 그동안 배운 AI를 로봇과 연동 시키기 위해 호환성을 위해 같은 프로그래밍 언어로 구성하면 유리한 부분이 있다.

또한 시스템을 구축한 후 유지/보수/관리 측면에서 다른 프로그래밍 언어로 구축하게 되면 인력 관리에 이슈가 생길 수 있어 하나의 프로그래밍 언어로 통일하는 것이 좋다

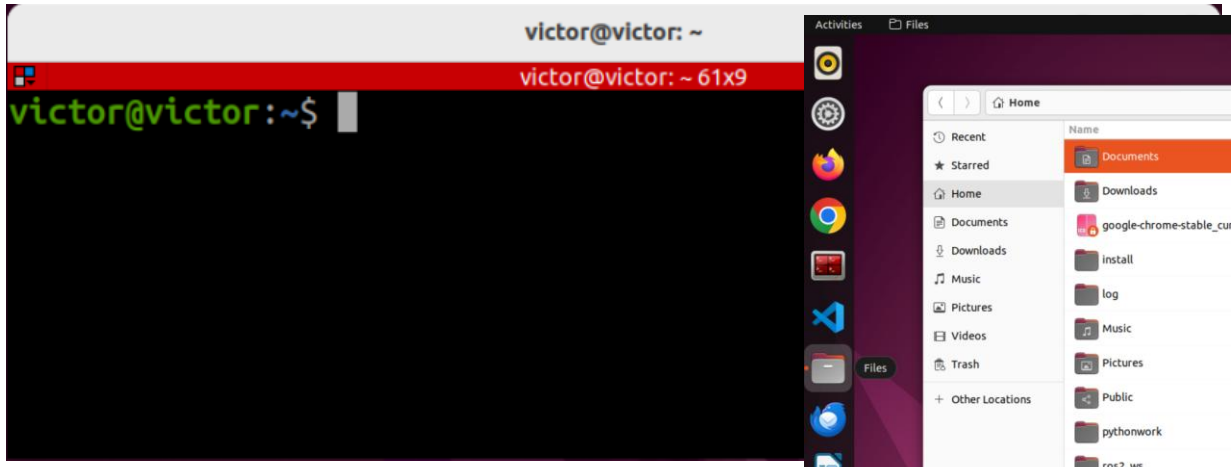
리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

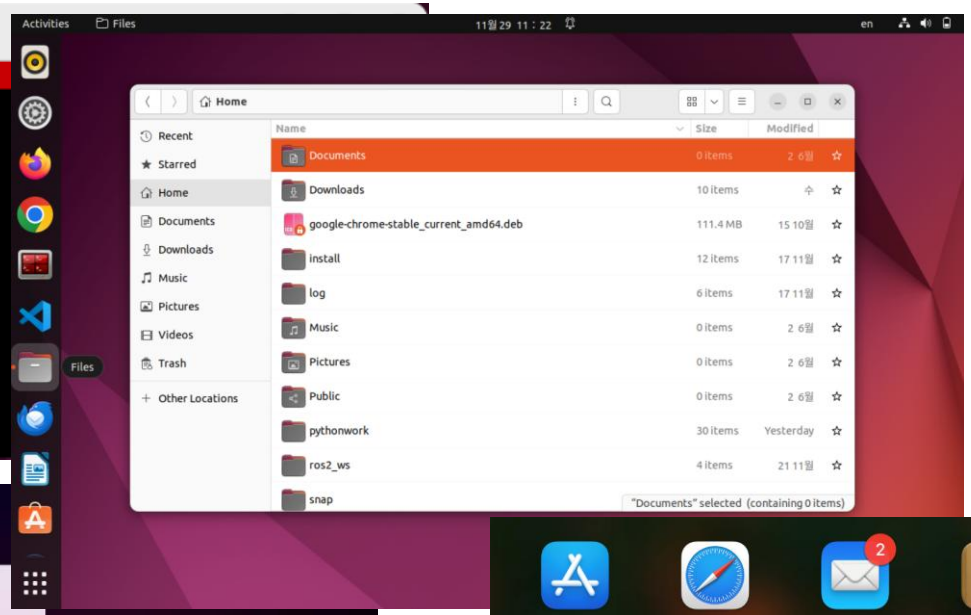


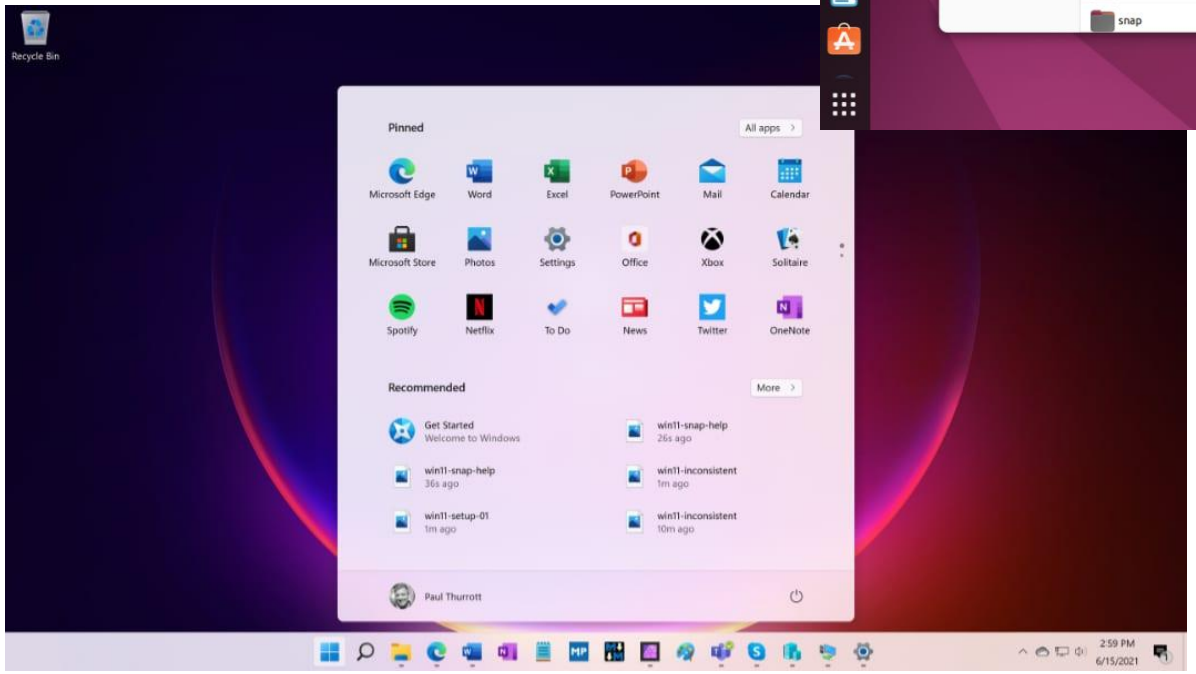
	GUI	CLI
방식	그래픽	텍스트
메모리 소비	높음	낮음
속도	느림	빠름
진입장벽	쉬움	어려움

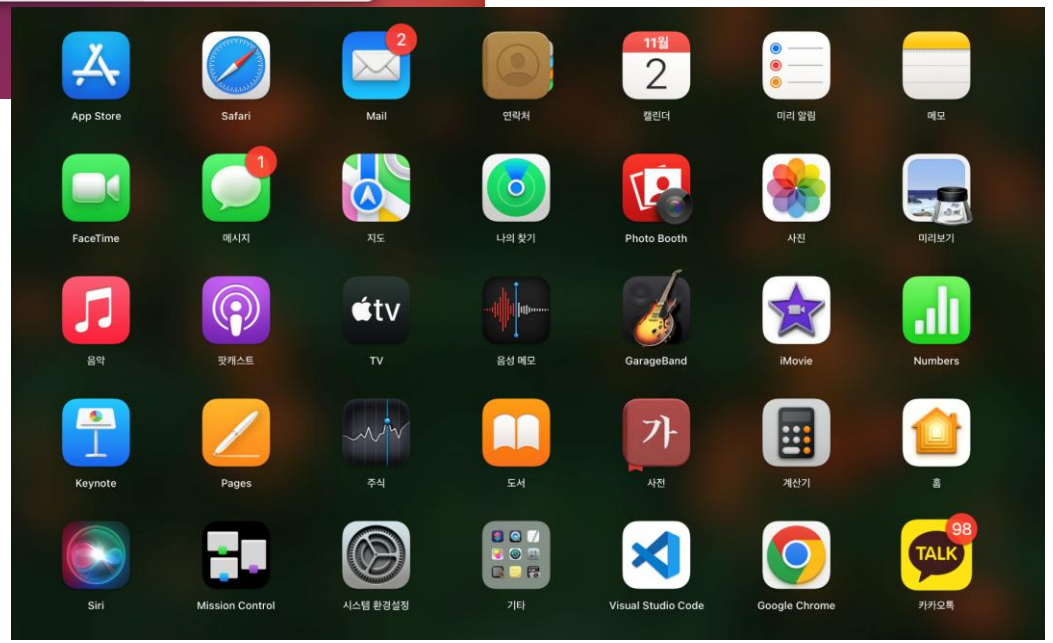
명령어	설명	
pwd	현재 나의(디렉토리) 위치 출력	- print working directory - 디렉토리 = 폴더
ls	현재 위치에서 디렉토리, 파일 목록 확인	list
ls -a	숨김파일까지 확인 가능한 옵션	all
ls -l	파일/디렉토리 정보를 자세히 표시하는 옵션	ls -al: 옵션은 동시에 여러 개 사용 가능
cd [이름]	해당 디렉토리로 이동	- change directory cd . : 현재 작업중인 디렉토리 cd .. : 현재 디렉토리의 상위 디렉토리 cd ~ : 홈 디렉토리로 이동
mkdir [이름]	디렉토리(폴더)를 생성	- make directory
rmdir [이름]	디렉토리(폴더)를 제거	- remove directory
clear	CLI 출력 내용 지우기	
history	내가 입력한 명령어 이력을 출력	공백 뒤에 숫자 값을 입력하면 가장 최근에 입력한 n개의 명령어 출력
touch [이름].[확장자]	새 파일을 생성	ex) touch index.html ->index.html파일생성 touch html/index.html ->html폴더안에 index.html파일생성
cat [이름].[확장자]	파일 내용 확인	-concatenate ex) cat html/index.html -> html폴더안에 index.html파일내용확인

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

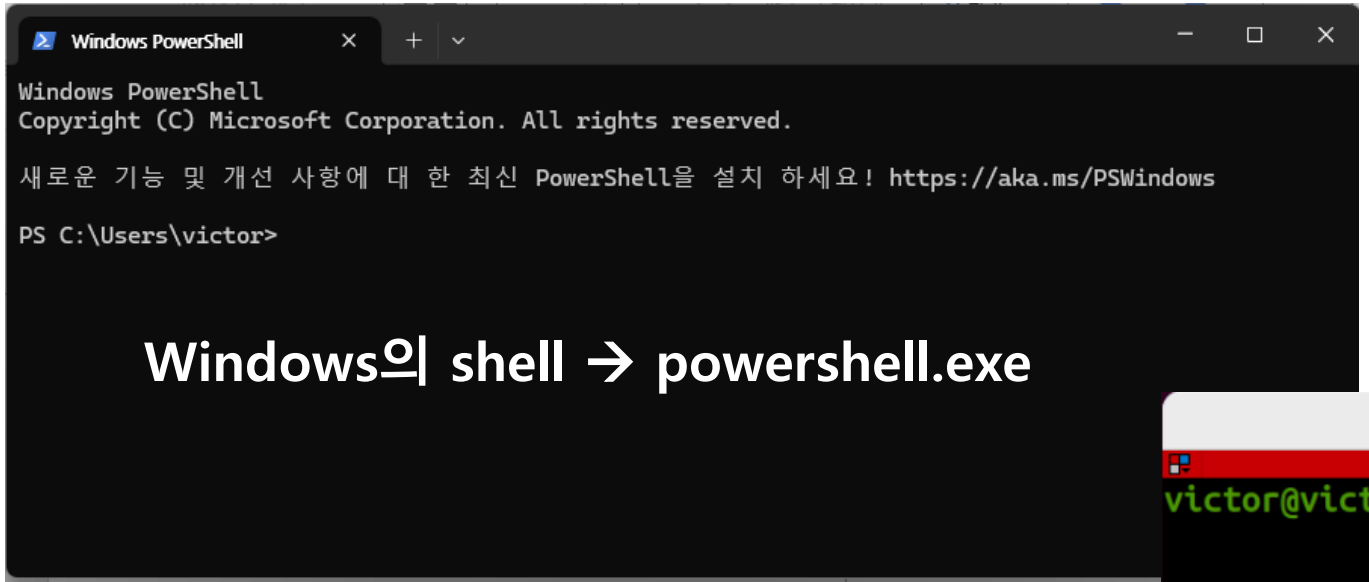




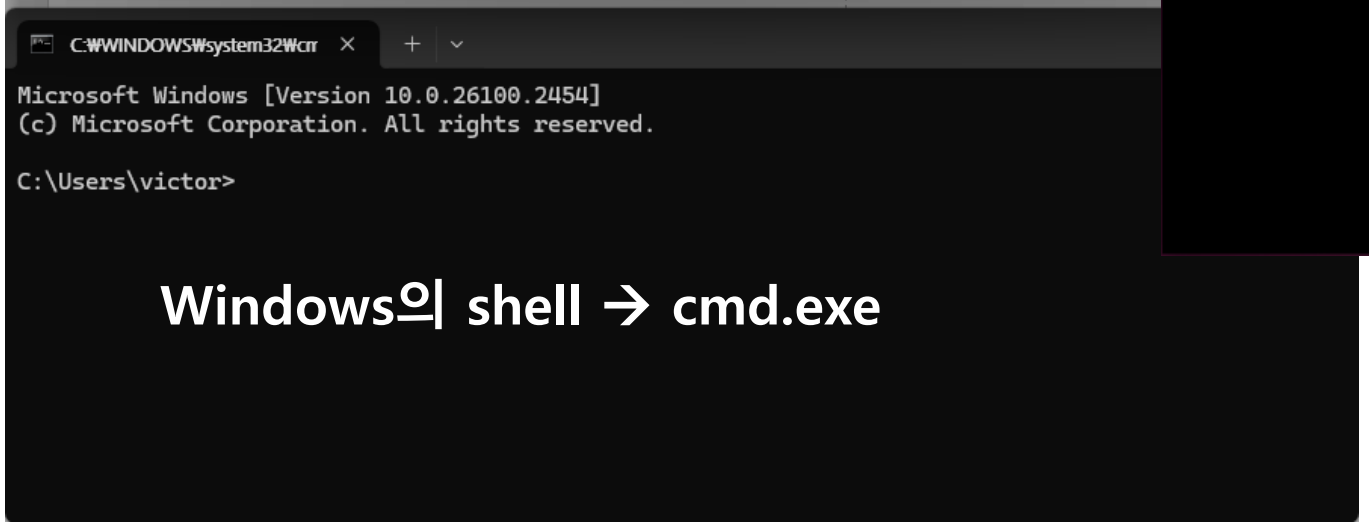




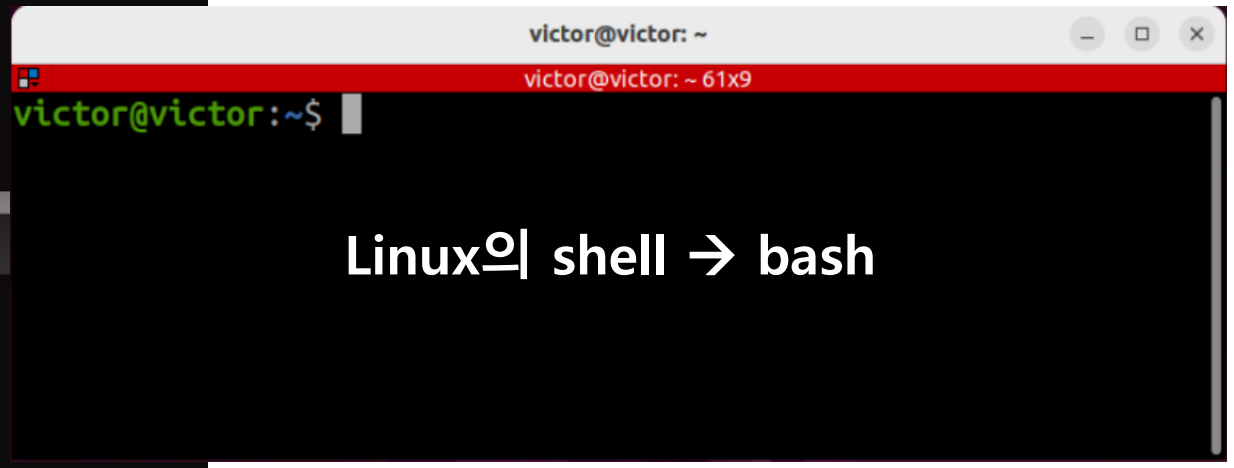
리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)



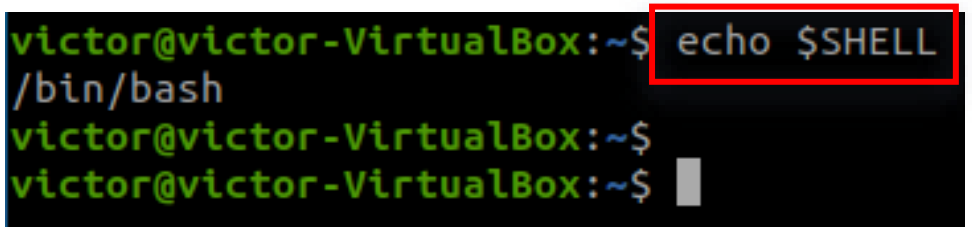
Windows의 shell → powershell.exe



Windows의 shell → cmd.exe

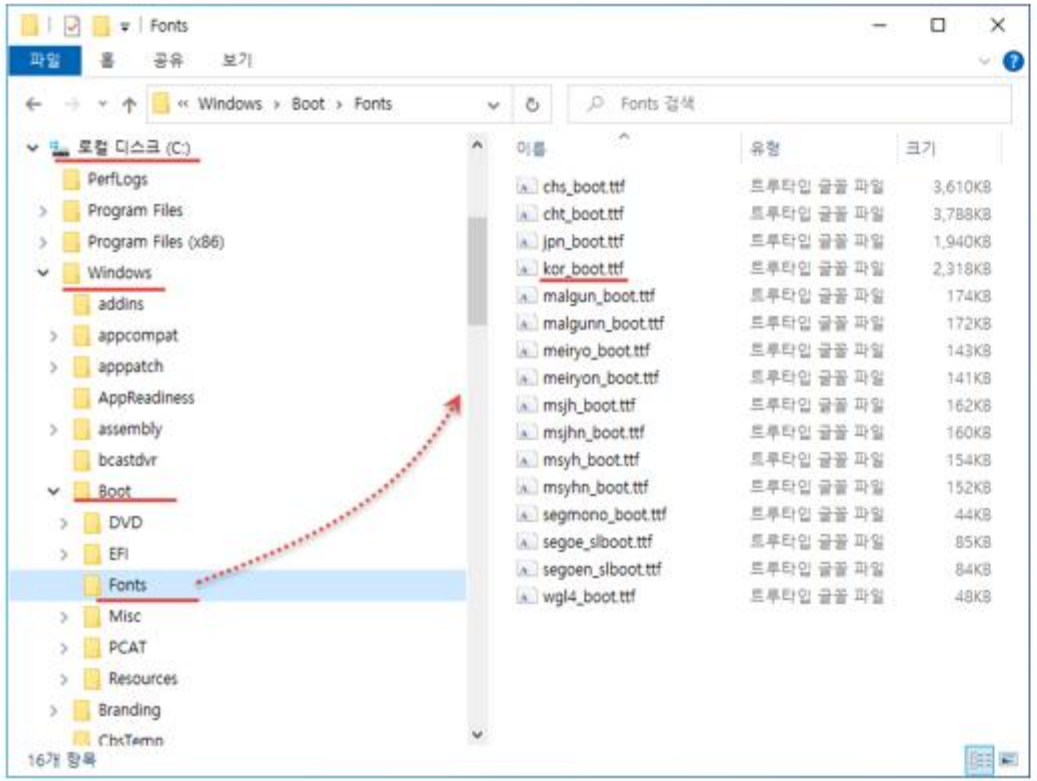


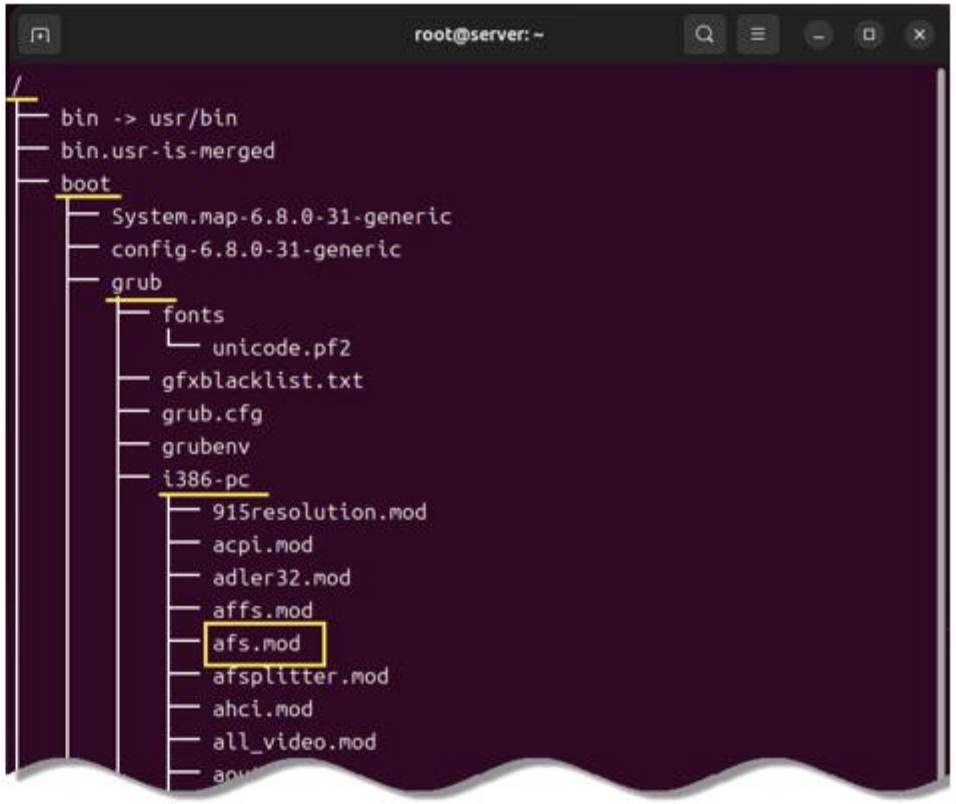
Linux의 shell → bash



리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

Windows와 리눅스의 디렉토리 구조 비교



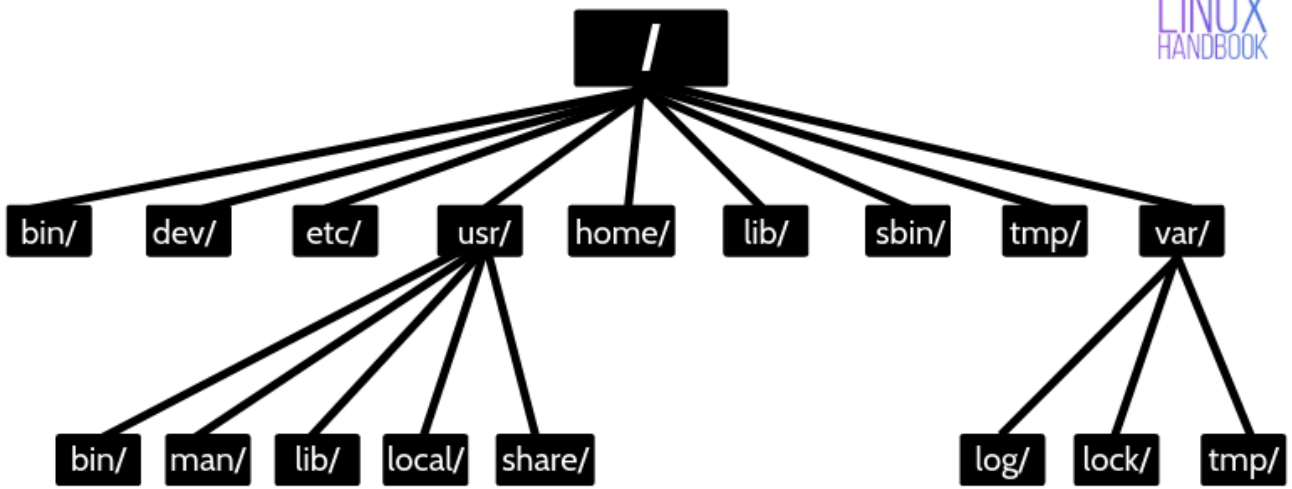


C:\Windows\Boot\Fonts\kor_boot.ttf

/boot/grub/i386-pc/afs.mod

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

FHS(Filesystem Hierarchy Standard) 구성



/bin: ls, cd, cp, mv과 같은 기본적인 명령어(binary)를 저장하는 디렉토리로, 대부분의 실행파일을 포함함

/boot: OS 부팅에 대한 파일을 담고 있는 디렉토리로, 커널 이미지 파일은 부팅 시 매우 중요함

/dev: 입출력 장치와 관련된 device 디렉토리

Ex) /dev/had(하드디스크), /dev/sda(SCSI 타입 하드디스크)

/etc: 시스템 환경 설정 파일과 시스템 부팅, 셧다운 시 필요한 파일들의 디렉토리

/home: user의 홈 디렉토리, 사용자 계정명과 동일하며 root는 /root가 홈 디렉토리

/media: CD_ROM,USB 등 외부 장치 연결 디렉토리

/mnt: 파일을 임시로 연결(mounting)하는 디렉토리

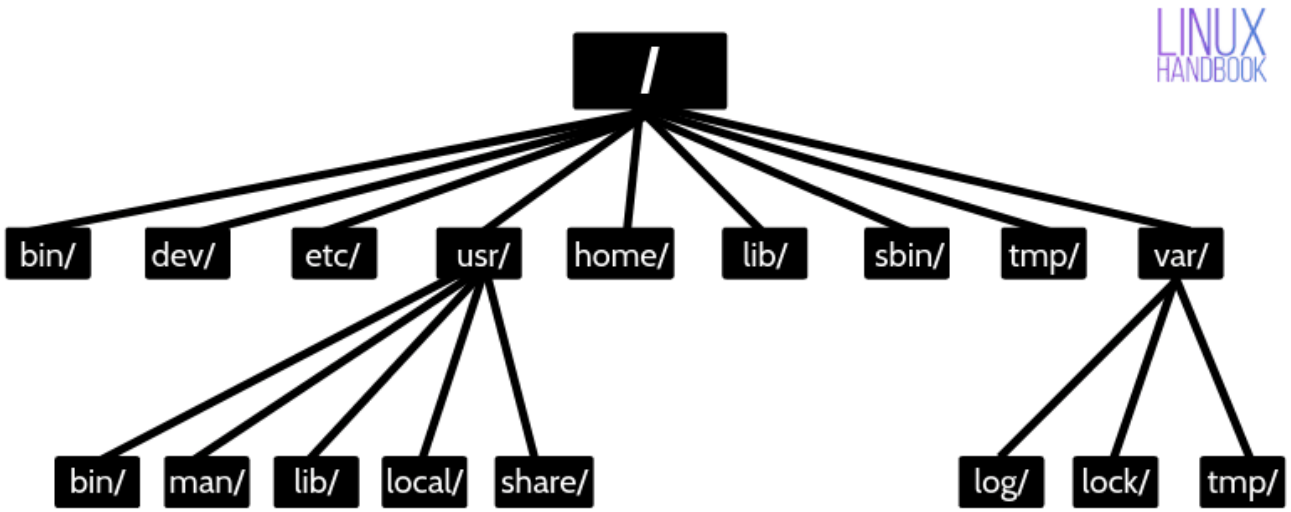
/proc: 프로세스(process)와 OS 정보를 제공하기 위한 가상 파일 시스템의 디렉토리로, 각종 정보를 kernel 모드가 아닌 user 모드에서 쉽게 접근할 수 있도록 해줌

-문자 디렉토리는 시스템과 커널 정보,숫자 디렉토리는 현재 실행 중인 프로세스의 정보를 나타냄

/root: 일반 사용자가 접근할 수 없는 시스템 관리자 root의 홈 디렉토리

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

FHS(Filesystem Hierarchy Standard) 구성



/sbin: system Binary를 의미하고 시스템 관리를 위한 실행 유틸리티를 담고 있다. Root 만이 실행할 수 있는 프로그램과 명령어가 있다. Ex) fdisk, reboot

/sys: 리눅스 kernel 관련 정보가 있는 디렉토리

/temp: 발생한 임시데이터가 저장되는 디렉토리, 수시로 생성 및 삭제되며 부팅 시 초기화됨

/usr: 기본 실행파일, 라이브버리 파일, 헤더 파일 등이 저장되어 있는 공유 파일 시스템 디렉토리로 사용자와 관련된 대부분의 응용프로그램과 파일이 저장되어있음

/var: 시스템 운영 중 발생한 가변 데이터와 로그가 저장되는 디렉토리

/opt: operation을 의미하며 타사 응용 프로그램을 설치하는 디렉토리, CentOS는 없음

/lib: 시스템 운영 및 프로그램 작동 시 필요한 공유 라이브러리(*.so)

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

루트 디렉토리

- 디렉토리(directory): 파일 시스템을 계층화할 때 사용하는 도구
- 루트 디렉토리(root directory): 파일 시스템의 최상단에 위치하는 디렉토리

현재 작업 디렉토리

- 현재 작업 디렉토리: Bash가 실행 중인 디렉터리

홈 디렉토리

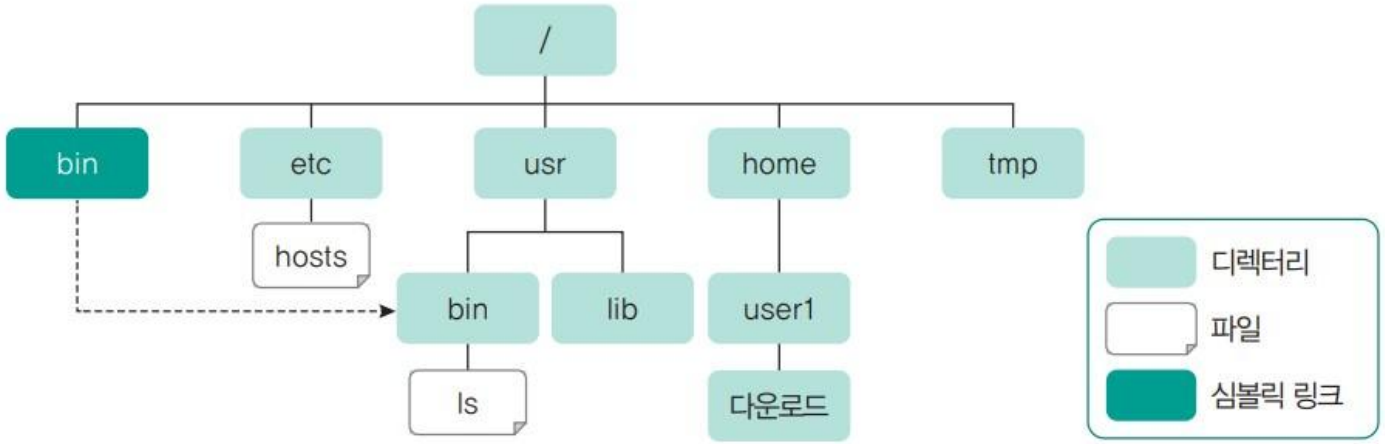
- 홈 디렉토리(home directory): 리눅스에 사용자를 추가하면 사용자별로 할당하는 디렉토리

▶ 루트 디렉터리 하위의 주요 디렉토리

디렉터리	용도	디렉터리	용도
/bin	기본 실행 명령어가 위치합니다.	/proc	procfs가 마운트됩니다.
/dev	디바이스 파일이 위치합니다.	/root	root 사용자의 홈 디렉터리입니다.
/etc	시스템 설정 파일이 위치합니다.	/sbin	시스템 관리용 명령어가 위치합니다.
/home	사용자의 홈 디렉터리가 위치합니다.	/sys	sysfs가 마운트됩니다.
/lib	시스템 라이브러리가 설치됩니다.	/tmp	시스템 임시 파일이 위치합니다.
/mnt	시스템에 디스크 등을 임시로 마운트할 때 사용합니다.	/usr	시스템 소프트웨어가 설치됩니다.

루트 디렉터리와 서브 디렉터리

- 최상단에 루트 디렉터리 (/)가 있고, 그 아래에 etc, usr, home, tmp 같은 디렉터리가 있음
- 루트 디렉터리 : 유일하게 부모 디렉터리가 없는 디렉터리. 아래에는 기본적으로 서브 디렉터리가 있음
- 디렉터리 아래에 있는 디렉터리를 서브 디렉터리sub directory 또는 하위 디렉터리라고 함
- 서브 디렉터리의 입장에서 보면 위에 자신을 포함하고 있는 디렉터리가 있는데, 이를 부모 디렉터리parent directory 또는 상위 디렉터리라고 함
- 상위 디렉터리는 ..(마침표 두 개)로 표시하며, .(마침표 한 개)는 현재 디렉터리를 말함



리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

디렉터리	기능
dev	장치 파일이 담긴 디렉터리
home	사용자 홈 디렉터리가 생성되는 디렉터리
media	DVD나 CD, USB 같은 외부 장치를 마운트(연결)하는 디렉터리
opt	추가 패키지가 설치되는 디렉터리
root	root 계정의 홈 디렉터리. 루트(/) 디렉터리와 다른 것이므로 혼동하지 않도록 한다.
sys	리눅스 커널과 관련된 파일이 있는 디렉터리
usr	기본 실행 파일과 라이브러리 파일, 헤더 파일 등 많은 파일을 가지고 있다. usr는 'Unix System Resource'의 약자
boot	부팅에 필요한 커널 파일을 가지고 있다.
etc	리눅스 설정을 위한 각종 파일을 가지고 있다.
lost+found	파일 시스템에 문제가 발생하여 복구할 경우, 문제가 되는 파일이 저장되는 디렉터리로 보통은 비어 있다.
mnt	파일 시스템을 임시로 마운트하는 디렉터리
proc	프로세스 정보 등 커널 관련 정보가 저장되는 디렉터리
run	실행 중인 서비스와 관련된 파일이 저장된다.
srv	FTP나 Web 등 시스템에서 제공하는 서비스의 데이터가 저장된다.
tmp	시스템 사용 중에 발생하는 임시 데이터가 저장된다. 시스템을 재시작하면 이 디렉터리에 있는 파일은 모두 삭제된다.
var	시스템 운영 중에 발생하는 데이터나 로그 등 내용이 자주 바뀌는 파일이 주로 저장된다.

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

현재 디렉터리 확인(pwd)

- 현재 디렉터리를 확인하는 명령은 pwd
- user1 계정으로 로그인하면 현재 디렉터리는 user1 계정의 홈 디렉터리가 됨

디렉터리 이동(cd)

- 디렉터리에서 다른 디렉터리로 이동할 때는 cd 명령을 사용
- cd 명령과 함께 이동하고자 하는 목적지 디렉터리를 지정하면 해당 디렉터리로 이동
- 이동할 디렉터리의 경로명으로 절대 경로명과 상대 경로명 둘 다 사용할 수 있음
- 이동한 뒤 pwd 명령을 사용하여 현재 디렉터리가 바뀌었는지 확인 (~가 tmp로 바뀜)
- 프롬프트에 현재 디렉터리의 이름을 표시하도록 설정되어 있는 것
- 상대 경로명을 이용하여 디렉터리를 이동할 경우 상위 디렉터리로 이동해야 하므로 ..(마침표 두 개)로 시작

원래의 홈 디렉터리로 이동방법

- cd /home/user1: 절대 경로명을 사용하여 홈 디렉터리로 이동
- cd ../../home/user1: 현재 /usr/lib 디렉터리에 있으므로 이를 기준으로 상대 경로명을 사용하여 홈 디렉터리로 이동
- cd ~: 홈 디렉터리를 나타내는 기호인 ~를 사용하여 홈 디렉터리로 이동
- cd: 목적지를 지정하지 않고 cd 명령만 사용하면 해당 계정의 홈 디렉터리로 이동

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

디렉터리 내용 확인

- ls : 디렉터리에 있는 파일이나 서브 디렉터리 등 디렉터리의 내용을 보는 명령
- ls 명령 은 다양한 기능을 제공하는 옵션을 사용하고, 내용을 보고 싶은 목적지 디렉터리를 인자로 지 정할 수 있다

ls

- 기능 디렉터리의 내용을 출력한다.
- 형식 ls [옵션] [디렉터리(파일)]
- 옵션 -a: 숨김 파일을 포함하여 모든 파일의 목록을 출력한다.
  -d: 디렉터리 자체의 정보를 출력한다.
  -i: 첫 번째 행에 inode 번호를 출력한다.
  -l: 파일의 상세 정보를 출력한다.
  -A: .(마침표)와 ..(마침표 두 개)를 제외한 모든 파일 목록을 출력한다.
  -F: 파일의 종류를 표시한다(*: 실행 파일, /: 디렉터리, @: 심볼릭 링크).
  -L: 심볼릭 링크 파일의 경우 원본 파일의 정보를 출력한다.
  -R: 서브 디렉터리의 목록까지 출력한다.
- 사용 예 ls ls -F ls -al /tmp

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

숨김 파일 확인하기: -a 옵션

- 리눅스에서는 파일명이나 디렉터리명을 .(마침표)로 시작하면 숨김 파일이 됨
- ls 명령만 사용해서는 보이지 않고 -a 옵션을 지정해야 함
- 현재 디렉터리를 나타내는 .(마침표)와 상위 디렉터리를 나타내는 ..(마침표 두 개)도 확인할 수 있음

파일의 종류 표시하기: -F 옵션

- ls 명령 에서도 -F 옵션을 사용하면 파일의 종류를 구분하는 기호가 표시 됨
- 파일명 뒤에 /가 붙으면 디렉터리, @이 붙으면 심볼릭 링크, *가 붙으면 실행 파일을 의미하고, 아무 표시도 없으면 일반 파일

옵션 여러 개 사용하기

- 옵션을 연결할 때는 -(하이픈) 뒤에 옵션만 나열
- 숨김 파일을 보여주는 a 옵션과 파일의 종류를 보여주는 F 옵션을 연결하여 사용하면 숨김 파일의 종류도 알 수 있음
- .(마침표)와 ..(마침표 두 개)에도 /가 붙어 있음

```
[user1@localhost ~]$ ls -aF
./  .bash_history .bash_profile .cache/  .local/  .viminfo 다운로드/ 바탕화면/ 사진/  음악/
../ .bash_logout  .bashrc       .config/ .mozilla/        공개/     문서/     비디오/ 서식/
```

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

지정한 디렉터리의 내용 출력하기

- 해당 디렉터리로 이동하지 않고도 디렉터리의 내용을 확인할 수 있음
- 옵션과 인자를 함께 사용할 수도 있음

상세 정보 출력하기: -l 옵션

- 디렉터리에 있는 파일들의 상세한 정보를 보려면 -l 옵션을 사용

디렉터리의 자체 정보 확인하기: -d 옵션

- 디렉터리의 자체 정보를 확인할 때는 -d 옵션을 사용

ls 명령과 비슷한 명령: dir, vdir

- 디렉터리의 내용을 보는 dir과 vdir 명령

```
[user1@localhost ~]$ ls -l
합계 0
drwxr-xr-x. 2 user1 user1 6  7월  9 10:48 공개
drwxr-xr-x. 2 user1 user1 6  7월  9 10:48 다운로드
drwxr-xr-x. 2 user1 user1 6  7월  9 10:48 문서
(생략)
```

필드 번호	필드 값	의미
1	d	다음과 같은 파일 종류를 나타낸다. –: 일반(정규) 파일 d: 디렉터리 파일 l: 심볼릭 링크 파일 b: 블록 단위로 읽고 쓰는 블록 장치 파일 c: 섹터 단위로 읽고 쓰는 문자 장치 파일 p: 파이프 파일(프로세스 간 통신에 사용되는 특수 파일) s: 소켓(네트워크 통신에 사용되는 특수 파일)
2	rwxr–xr–x	파일 접근 권한. 파일의 소유자, 그룹, 기타 사용자가 읽고 수정하고 실행할 수 있는 권한이 어떻게 부여되어 있는지를 보여준다.
3	2	하드 링크의 개수
4	user1	파일 소유자
5	user1	파일이 속한 그룹
6	6	파일 크기(바이트 단위)
7	7월 9 10:48	파일이 마지막으로 수정된 시간
8	공개	파일명

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

디렉터리 한 개 만들기

- 디렉터리를 한 개만 만들려면 mkdir 명령에 인자로 생성하려는 디렉터리를 지정하면 됨

```
[user1@localhost ~]$ mkdir temp
[user1@localhost ~]$ ls
temp  공개  다운로드  문서  바탕화면  비디오  사진  서식  음악
```

동시에 디렉터리 여러 개 만들기

```
[user1@localhost ~]$ mkdir tmp1 tmp2 tmp3
[user1@localhost ~]$ ls
temp  tmp1  tmp2  tmp3  공개  다운로드  문서  바탕화면  비디오  사진  서식  음악
```

중간 디렉터리를 자동으로 만들기: -p 옵션

- mkdir 명령 다음에 -p 옵션을 사용하면, 생성할 디렉터리로 지정한 경로 중 중간 단계의 디렉터리가 없을 경우 자동으로 중간 단계 디렉터리를 생성한 후 최종 디렉터리를 만듦
- 비교해보기 : mkdir 명령에 -p 옵션을 사용하지 않은 경우 vs mkdir 명령에 -p 옵션을 사용한 경우

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

디렉터리 삭제

- mkdir 명령 예에서 만든 tmp3을 삭제하는 예
- rmdir 명령으로 디렉터리를 삭제할 때는 해당 디렉터리가 비어 있어야 함
- 디렉터리에 파일이나 서브 디렉터리가 남아 있으면 rmdir로 디렉터리를 삭제할 수 없음
- 비어 있지 않은 디렉터리를 삭제하려 했을 때

실습

- 홈 디렉터리로 이동
- Test 디렉터리 만들고 이동하기
- 디렉터리 동시에 만들기 $mkdir one two three
- 중간경로 tmp 디렉터리 자동생성 $mkdir –p one/tmp/test
- 하위 디렉터리보기 $ls –R one
- $rmdir one
- $rmdir two three 실행해보기

파일 내용 출력

- 파일 내용을 연속으로 출력하기: cat

파일 내용 출력

- /etc/hosts 파일의 내용을 cat 명령으로 확인

파일 내용을 화면 단위로 출력하기: more

- more 명령은 파일 내용을 화면 단위로 출력하고, 출력할 내용이 더 있으면 화면 하단에 '--More--(0%)'와 같이 알려줌

```
[user1@localhost ~]$ more /etc/services
# /etc/services:
# $Id: services,v 1.49 2017/08/18 12:43:23 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2016-07-08
(생략)
chargen         19/udp          ttytst source
ftp-data        20/tcp
--More--(0%)
```

파일 내용을 화면 단위로 출력하기: less

- more 명령은 이미 스크롤되어 지나간 내용을 다시 볼 수 없다는 것
- less 명령을 사용하면 파일 내용을 앞뒤로 스크롤하며 이동할 수 있음

less

- 기능 파일 내용을 화면 단위로 출력한다.
- 형식 less [파일]
- 사용 예 less file1

키	동작
j, ↓	한 행씩 다음 행으로 스크롤한다.
k, ↑	한 행씩 이전 행으로 스크롤한다.
Space Bar, Ctrl+f	다음 화면으로 이동한다.
Ctrl+b	이전 화면으로 이동한다.

파일 내용의 뒷부분 출력하기: tail

- tail은 파일 뒷부분의 몇 행을 출력, 기본값은 10으로 파일 뒷부분의 10행이 출력 됨

```
[user1@localhost ~]$ tail /etc/services
aigairserver    21221/tcp        # Services for Air Server
ka-kdp          31016/udp        # Kollective Agent Kollective Delivery
ka-sddp         31016/tcp        # Kollective Agent Secure Distributed Delivery
edi_service     34567/udp        # dhanalakshmi.org EDI Service
axio-disc       35100/tcp        # Axiomatic discovery protocol
axio-disc       35100/udp        # Axiomatic discovery protocol
pmwebapi        44323/tcp        # Performance Co-Pilot client HTTP API
cloudcheck-ping 45514/udp        # ASSIA CloudCheck WiFi Management keepalive
cloudcheck      45514/tcp        # ASSIA CloudCheck WiFi Management System
spremotetablet  46998/tcp        # Capture handwritten signatures
```

리눅스의 시간(2038년 1월 19일 03시 14분 07초)

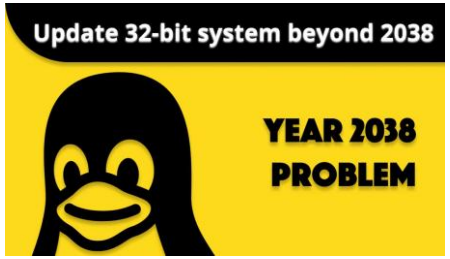
1. 제목 : 2038년 문제(Y2038)
2. 개요 :

   32bit 운영체계를 사용하는 OS(Linux, Unix)는 2038/01/19 03:14:07초를 지나게 되면 1901/12/31 혹은 1970/01/01 시점으로 타임슬립하는 문제.

   정식명칭은 Y2K38, Y2038이라고 함
3. 원인 :

   컴퓨터에서 그레고리력 시간을 계산하는 방법에는 여러 가지가 있는데, 현재 보편적인 방법은 Unix Time을 사용함. 해당 방법은 32bit 크기의 정수형을 사용하여 시간을 나타냄. 초당 1씩 증가. 32bit의 한계점은 2,147,483,647이므로 해당 수를 넘어가게 되면 overflow현상이 발생하고, 최소값으로 돌아가게 됨
4. 해결 방법 :
   - 부호 없는 정수형으로 변경. 음수를 제외하면 0 ~ 4,294,967,295까지 증가. 즉, 2106년까지 늦출 수 있다.
   - 그러나 1970년 1월 1일 이전의 시간을 셀 수 없으므로 그 이전 출생자들의 정보가 모두 사라지는 단점
   - OS를 64bit이상으로 변경. 단순히 OS만 변경하면 안되고 32bit에 맞춰져 있는 실행파일, 라이브러리 등 64bit정수형으로 변경필요
5. 이 문제를 해결하면 언제까지 가능?

   64bit 정수형 최대값 = 9,223,372,036,854,775,807. 1970년부터 계산하면 서기 2922억7702만6596년 12월 4일 15시 30분 8초
6. 참고 :
   - Y2K : 1999/12/31 23:59:59 → 2000/01/01 00:00:00
   - 10년 = 10년 x 365일 x 24시간 x 3600초 = 315,360,000초(약 3억초)
   - 100년 = 100년 x 365일 x 24시간 x 3600초 = 3,153,600,000초(약 31억초)
   - 80년 = 80년 x 365일 x 24시간 x 3600초 = 2,522,880,000초(약 25억초)



리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

root

시스템을 관리할 수 있는 관리자 권한의 계정이자 슈퍼 유저
리눅스 파일 체제의 최상위 디렉토리(/)로도 표현한다.

root 권한이 있으면 모든 파일과 디렉토리에 대해 읽고 쓸 수 있고, 생성할 수도 있지만 제거할 수도 있다.
시스템 구성을 변경할 수도 있다. 그래서 매우 편하지만 조심히 행동해야 하는 계정이다.

su (Switch User)

현재 계정을 로그아웃하지 않고 다른 계정으로 전환하는 명령어

sudo란 무엇인가?

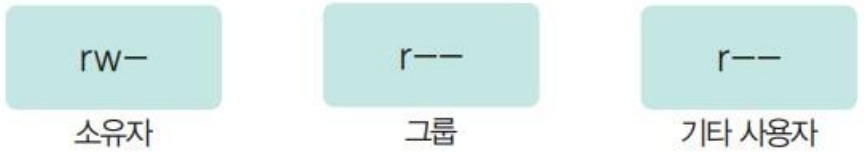
sudo는 superuser do의 줄임말로, 일시적으로 관리자 권한을 부여하여 특정 명령어를 실행할 수 있게 해주는 명령어입니다.
리눅스에서는 기본적으로 일반 사용자와 관리자(root) 사용자를 구분하여 시스템을 보호합니다. sudo를 통해 사용자는 root
계정의 권한을 일시적으로 사용할 수 있으며, 시스템 파일을 수정하거나 프로그램을 설치하는 등 관리자 권한이 필요한
작업을 수행할 수 있습니다

권한 설정 이해하기(drwxr-xr-x 형식)

- 리눅스에서 파일과 디렉터리는 세 가지 사용자(소유자, 그룹, 기타 사용자)에게 읽기(r), 쓰기(w), 실행(x) 권한을 가질 수 있습니다. 이 권한은 chmod 명령어를 통해 수정 가능합니다.

형식 설명 (drwxr-xr-x)

- 첫 글자: 파일 타입 (d는 디렉터리, -는 파일)
- 그 다음 세 글자: 소유자의 권한 (rwx - 읽기, 쓰기, 실행)
- 그 다음 세 글자: 그룹의 권한 (r-x - 읽기, 실행)
- 마지막 세 글자: 기타 사용자의 권한 (r-x - 읽기, 실행)



접근 권한	의미
rwxr-xr-x	소유자는 읽기 · 쓰기 · 실행 권한을 모두 가지고, 그룹과 기타 사용자는 읽기 · 실행 권한을 가지고 있다.
r-xr-xr-x	소유자, 그룹, 기타 사용자 모두 읽기 · 실행 권한을 가지고 있다.
rw-------	소유자만 읽기 · 쓰기 권한을 가지고, 그룹과 기타 사용자에게는 아무 권한이 없다.
rw-rw-rw-	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 권한을 가지고 있다.
rwxrwxrwx	소유자, 그룹, 기타 사용자 모두 읽기 · 쓰기 · 실행 권한을 가지고 있다.
rwx------	소유자만 읽기 · 쓰기 · 실행 권한을 가지고, 그룹과 기타 사용자에게는 아무 권한이 없다.
r-------	소유자만 읽기 권한을 가지고 있다.

파일 권한 수정 명령어

- 읽기, 쓰기, 실행 권한 부여
- chmod u+r 파일명: 소유자에게 읽기 권한 추가
- chmod g-w 파일명: 그룹의 쓰기 권한 제거
- chmod o+x 파일명: 기타 사용자에게 실행 권한 추가
- 숫자 코드로 설정
- 4 = 읽기 (r), 2 = 쓰기 (w), 1 = 실행 (x)
- 예를 들어, chmod 755 파일명 명령어는 drwxr-xr-x와 같은 설정을 의미합니다.

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

```
victor@victor-VirtualBox: ~ 140x39
victor@victor-VirtualBox:~$ ll
total 112
drwxr-x--- 23 victor victor 4096  3월 18 22:25 ./
drwxr-xr-x  3 root   root   4096  2월 25 18:00 ../
-rw-------  1 victor victor 4076  3월 22 12:27 .bash_history
-rw-r--r--  1 victor victor  220  2월 25 18:00 .bash_logout
-rw-r--r--  1 victor victor 4173  3월 18 19:02 .bashrc
drwx------ 18 victor victor 4096  3월 22 07:52 .cache/
drwx------ 23 victor victor 4096  3월 22 07:52 .config/
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Desktop/
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Documents/
drwxrwxr-x  3 victor victor 4096  2월 27 16:21 .dotnet/
drwxr-xr-x  3 victor victor 4096  3월 22 12:07 Downloads/
drwx------  2 victor victor 4096  3월 16 19:57 .gnupg/
drwx------  3 victor victor 4096  2월 25 18:13 .local/
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Music/
drwxrwxr-x  7 victor victor 4096  3월 18 22:26 opencv_
drwxr-xr-x  3 victor victor 4096  3월  2 12:35 Picture
drwx------  3 victor victor 4096  2월 26 16:40 .pki/
-rw-r--r--  1 victor victor  807  2월 25 18:00 .profil
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Public/
drwxrwxr-x  4 victor victor 4096  2월 27 12:01 .ros/
drwxrwxr-x  6 victor victor 4096  2월 27 11:58 ros2_ex
drwxrwxr-x  8 victor victor 4096  3월 22 07:20 ros2_ws/
drwx------  5 victor victor 4096  2월 26 16:33 snap/
drwx------  2 victor victor 4096  3월 16 19:19 .ssh/
-rw-r--r--  1 victor victor    0  2월 26 16:15 .sudo_as_admin_successful
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Templates/
drwxr-xr-x  2 victor victor 4096  2월 25 18:13 Videos/
drwxrwxr-x  4 victor victor 4096  2월 26 17:39 .vscode/
victor@victor-VirtualBox:~$
```

```
drwxr-xr-x  3 victor victor 4096  3월 22 12:07 Downloads/
```

- `d` → 디렉토리(directory)
- `rwx` **(소유자** `u`**)** → 읽기(r), 쓰기(w), 실행(x)
- `r-x` **(그룹** `g`**)** → 읽기(r), 실행(x)
- `r-x` **(기타 사용자** `o`**)** → 읽기(r), 실행(x)

- `u=rwx` → 소유자에게 **읽기, 쓰기, 실행 권한** 부여
- `g=rx` → 그룹에게 **읽기, 실행 권한** 부여 (쓰기 권한 없음)
- `o=rx` → 기타 사용자에게 **읽기, 실행 권한** 부여

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

[chmod 사용하여 권한 변경]

```
drwxr-xr-x  3 victor victor 4096  3월 22 12:07 Downloads/
```

권한	8진수 값
`---` (000)	0
`--x` (001)	1
`-w-` (010)	2
`-wx` (011)	3
`r--` (100)	4
`r-x` (101)	5
`rw-` (110)	6
`rwx` (111)	7

1. 기호 모드(Symbolic Mode) 사용

```
chmod u=rwx,g=rx,o=rx mydir
chmod u+rwx,g+rx,o+rx mydir
chmod g-w,o-w mydir
```

- `u=rwx` → 소유자에게 **읽기, 쓰기, 실행 권한** 부여
- `g=rx` → 그룹에게 **읽기, 실행 권한** 부여 (쓰기 권한 없음)
- `o=rx` → 기타 사용자에게 **읽기, 실행 권한** 부여

2. 숫자 모드(Octal Mode) 사용

```
chmod 755 mydir
```

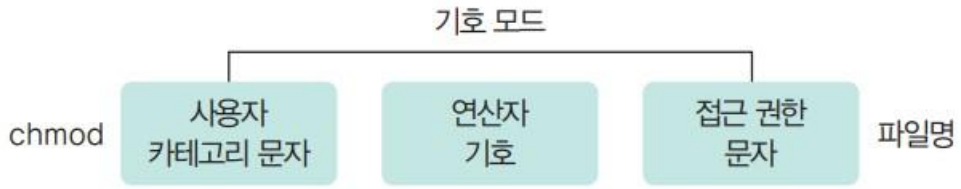
- `7 (rwx)` → 소유자(user)에게 읽기(r), 쓰기(w), 실행(x) 권한
- `5 (r-x)` → 그룹(group)에게 읽기(r), 실행(x) 권한 (쓰기 없음)
- `5 (r-x)` → 기타 사용자(others)에게 읽기(r), 실행(x) 권한 (쓰기 없음)

※ **하위 폴더와 파일에도 적용하는 옵션**

```
chmod -R 755 mydir
```

- `-R` (Recursive) 옵션은 **하위 폴더와 파일에도 동일한 권한을 적용**합니다.

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)



기호 모드

- 사용자 카테고리: 소유자, 그룹, 기타 사용자를 나타내는 문자로 표기
- 연산자: 권한 부여나 제거를 나타내는 기호로 표기
- 접근 권한 기호: 읽기, 쓰기, 실행을 나타내는 문자를 사용

권한 표기	의미
u+w	소유자(u)에게 쓰기(w) 권한 부여(+)
u-x	소유자(u)의 실행(x) 권한 제거(-)
g+w	그룹(g)에 쓰기(w) 권한 부여(+)
o-r	기타 사용자(o)의 읽기(r) 권한 제거(-)
g+wx	그룹(g)에 쓰기(w)와 실행(x) 권한 부여(+)
+wx	모든 사용자에게 쓰기(w)와 실행(x) 권한 부여(+)
a+rwx	모든 사용자에게 읽기(r), 쓰기(w), 실행(x) 권한 부여(+)
u=rwx	소유자(u)에게 읽기(r), 쓰기(w), 실행(x) 권한 설정(=)
go+w	그룹(g)과 기타 사용자(o)에게 쓰기(w) 권한 부여(+)
u+x,go+w	소유자(u)에게 실행(x) 권한을 부여하고(+) 그룹(g)과 기타 사용자(o)에게 쓰기(w) 권한 부여(+)

표 3-3 기호 모드에서 사용하는 문자와 기호

구분	문자/기호	의미
사용자 카테고리 문자	u	파일 소유자
	g	파일 소유 그룹
	o	소유자와 그룹 이외의 기타 사용자
	a	전체 사용자
연산자 기호	+	권한 부여
	-	권한 제거
	=	접근 권한 설정
접근 권한 문자	r	읽기 권한
	w	쓰기 권한
	x	실행 권한

루트 암호 설정 하기

$sudo passwd root

기호 모드로 접근 권한 변경하기

그룹에 쓰기와 실행 권한을 부여(g+wx)

```
[user1@localhost ch3]$ chmod g+wx test.txt
[user1@localhost ch3]$ ls -l test.txt
-r--rwxr--. 1 user1 user1 158  8월 13 20:57 test.txt
```

기타 사용자에게 실행 권한을 부여(o+x)

```
[user1@localhost ~]$ mkdir Test
[user1@localhost ~]$ cd Test
[user1@localhost Test]$
```

그룹과 기타 사용자의 실행 권한을 제거(go-x)

```
[user1@localhost ch3]$ chmod go-x test.txt
[user1@localhost ch3]$ ls -l test.txt
-r--rw-r--. 1 user1 user1 158  8월 13 20:57 test.txt
```

상대 경로와 절대 경로

경로(path): 파일의 위치 정보를 표현한 것

상대 경로(relative path): 현재 작업 디렉터리를 기준으로 파일 경로를 나타냄

터미널

```
gilbut@ubuntu2404:~/Downloads$ cd ../Pictures/
gilbut@ubuntu2404:~/Pictures$ pwd
/home/gilbut/Pictures
```

절대 경로(absolute path): 루트 디렉터리를 기준으로 파일 경로를 나타냄

터미널

```
gilbut@ubuntu2404:~$ cd /home/gilbut/Pictures/
gilbut@ubuntu2404:~/Pictures$ pwd
/home/gilbut/Pictures
```

파일 내용 검색하기 : grep

grep의 가장 기본적인 사용법으로 인자로 지정한 문자열을 검색하는 예

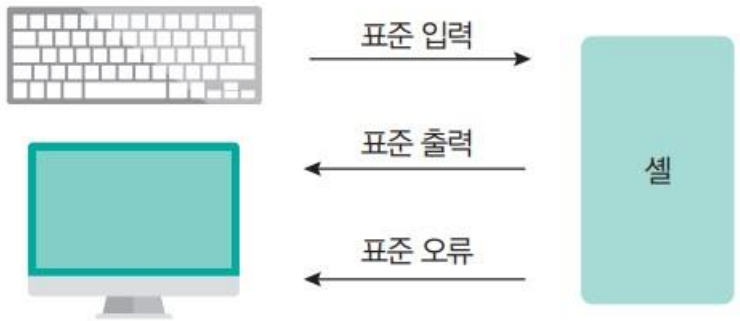
```
[user1@localhost ch2]$ grep DHCP /etc/services
dhcp-failover  647/tcp              # DHCP Failover
dhcp-failover  647/udp              # DHCP Failover
```

-n 옵션을 사용하면 검색된 행 번호도 함께 출력됨

```
[user1@localhost ch2]$ grep -n DHCP /etc/services
1413:dhcp-failover   647/tcp                # DHCP Failover
1414:dhcp-failover   647/udp                # DHCP Failover
```

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

표준 입출력 장치



- 표준 입력 장치: 셸이 작업을 수행하는 데 필요한 정보를 받아들이는 장치
- 표준 출력 장치: 실행 결과를 내보내는 장치
- 표준 오류 장치: 표준 출력과 별도로 오류 메시지를 내보내는 장치
- 표준 입력 장치는 키보드로 설정되어 있고, 표준 출력 및 표준 오류 장치는 화면으로 설정되어 있음

- 파일 디스크립터: 작업 중 필요한 파일에 일련번호를 붙여서 관리하는 값
- 입출력 장치를 변경할 때 파일 디스크립터를 사용

파일 디스크립터	파일 디스크립터 대신 사용하는 이름	정의
0	stdin	명령의 표준 입력
1	stdout	명령의 표준 출력
2	stderr	명령의 표준 오류

출력 리다이렉션

- 리다이렉션: 표준 입출력 장치를 파일로 바꾸는 것
- 출력 결과를 저장할 파일이 이미 존재하는 파일일 경우, 기존 파일의 내용을 유지할지 말지에 따라 달라짐
- 기존 파일의 내용을 삭제하고 새로 결과를 저장할 때는 >를, 기존 파일의 내용 뒤에 결과를 추가할 때는 >>를 사용

- **파일 덮어쓰기 : >**
  - 표준 출력 파일을 바꾸는 특수문자 >
  - 첫 번째 형식의 1은 파일 디스크립터 1번을 의미
  - 파일 디스크립터 1은 생략 가능하며, 보통 1이 생략된 두 번째 형식을 사용
  - 셸은 >를 사용한 리다이렉션에서 지정한 파일명의 파일이 없으면 파일을 생성하여 명령의 수행 결과를 저장
  - 해당 파일이 있으면 기존 내용이 없어지고 명령의 수행 결과로 대체되므로 출력 리다이렉션을 사용할 때는 먼저 해당 이름의 파일이 있는지 확인해야 함

```
>
• 기능  파일 리다이렉션(덮어쓰기)을 한다.
• 형식  명령 1> 파일명
        명령 > 파일명
```

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

```
victor@victor:~/Downloads$ cat > out1
Welcome ROKEY!
from DOOSAN ROBOTICS
victor@victor:~/Downloads$ ll
```

```
victor@victor:~/Downloads$ ll
total 133812
drwxr-xr-x  2 victor victor      4096 11월 29 13:21  ./
drwxr-x--- 27 victor victor      4096 11월 27 17:09  ../
-rw-rw-r--  1 victor victor 102126134 10월 21 21:24  code_1.94.2-1728494015_amd64.deb
-rw-rw-r--  1 victor victor       113 11월 17 22:11  my_first_node.py
-rw-rw-r--  1 victor victor       770 11월 17 21:41  my_first_publisher.py
-rw-rw-r--  1 victor victor       627 11월 17 20:49  my_first_subscriber.py
-rw-rw-r--  1 victor victor        36 11월 29 13:21  out1
-rw-rw-r--  1 victor victor    164466  6월  2 22:01  P20210317_181959481_6D1E8DEC-B6CD-4A46-A85A-1E
0E713D2D89.JPG
-rw-rw-r--  1 victor victor     13403 11월 27 17:08  PythonWork.zip
-rwxrwxr-x  1 victor victor      3082  6월 28 12:39  ros2-humble-desktop-main.sh*
-rw-rw-r--  1 victor victor  34666649 11월 17 16:03 '로봇운영체제 ROS 2 특강.pptx'
-rw-rw-r--  1 victor victor      1750 11월 15 18:03  ROS설치파일.zip
-rwxrwxr-x  1 victor victor       710  6월 28 12:39  tutorial.sh*
victor@victor:~/Downloads$ cat out1
Welcome ROKEY!
from DOOSAN ROBOTICS
victor@victor:~/Downloads$
```

리눅스 CLI 실습(디렉토리, 계정, 기본 명령어 등)

```
 1455  cd Downloads/
 1456  ll
 1457  cat > out1
 1458  ll
 1459  cat out1
 1460  cat >> out1
 1461  date >> out1
 1462  cat out1
 1463  history
victor@victor:~/Downloads$ !1458
ll
total 133812
drwxr-xr-x  2 victor victor      4096 11월 29 13:21  ./
drwxr-x--- 27 victor victor      4096 11월 27 17:09  ../
-rw-rw-r--  1 victor victor 102126134 10월 21 21:24  code_1.94.2-1728494015_amd64.deb
-rw-rw-r--  1 victor victor       113 11월 17 22:11  my_first_node.py
-rw-rw-r--  1 victor victor       770 11월 17 21:41  my_first_publisher.py
-rw-rw-r--  1 victor victor       627 11월 17 20:49  my_first_subscriber.py
-rw-rw-r--  1 victor victor        85 11월 29 13:24  out1
```

```
victor@victor:~/Downloads$ cat out1
Welcome ROKEY!
from DOOSAN ROBOTICS
victor@victor:~/Downloads$ cat >> out1
Welcome to ROS2
victor@victor:~/Downloads$ date >> out1
victor@victor:~/Downloads$ cat out1
Welcome ROKEY!
from DOOSAN ROBOTICS
Welcome to ROS2
2024. 11. 29. (금) 13:24:48 KST
victor@victor:~/Downloads$
```

사용법	기능
!!	바로 직전에 실행한 명령을 재실행한다.
!번호	히스토리에서 해당 번호의 명령을 재실행한다.
!문자열	히스토리에서 해당 문자열로 시작하는 마지막 명령을 재실행한다.

리눅스 CLI 실습

- **ls**

  Windows의 dir과 같은 역할로,해당 디렉터리에 있는 파일의 목록을 나열

  예) # ls /etc/systemd

형식 `ls [옵션] [파일]`

- **cd**

  디렉터리를 이동

  예) # cd ../etc/systemd

- **pwd**

  현재 디렉터리의 전체 경로를 출력

터미널

```
gilbut@ubuntu2404:~$ pwd
/home/gilbut
```

- **touch**

  크기가 0인 새 파일을 생성, 이미 존재하는 경우 수정 시간을 변경

  예) # touch abc.txt

옵션	설명
-a	모든 파일 출력
-l	파일의 여러 속성을 포함해 길게 출력
-t	생성된 순서로 파일 출력(오래된 파일이 가장 뒤로 감)
-R	하위 모든 디렉터리 순회
-h	사람이 읽기 좋은 크기(KB, MB, GB 등)로 파일 출력

리눅스 CLI 실습

- **rm**
  파일이나 디렉터리를 삭제
  예) # rm -rf abc

- **cp**
  파일이나 디렉터리를 복사
  예) # cp abc.txt cba.txt

- **mv**
  파일과 디렉터리의 이름을 변경하거나 위치 이동 시 사용
  예) mv abc.txt www.txt

- **mkdir**
  새로운 디렉터리를 생성
  예) # mkdir abc

실습 디렉터리 생성하기

- **mkdir**(make directory) 명령어

- 실습 순서
  1. ls 명령어로 animals 디렉터리 생성
  2. dog, cat, cow 디렉터리 생성
  3. snake 디렉터리 생성
  4. -p 옵션을 넣지 않고 fruits 디렉터리 하위에 apple 디렉터리 생성
  5. -p 옵션을 넣고 생성
  6. -p 옵션으로 fruits/apple 디렉터리를 다시 생성

리눅스 CLI 실습

- **rmdir**
  디렉터리를 삭제. (단, 비어 있어야 함)
  예) # rmdir abc

- **cat**
  텍스트로 작성된 파일을 화면에 출력
  예) # cat a.txt b.txt

- **head, tail**
  텍스트로 작성된 파일의 앞 10행 또는 마지막 10행만 출력
  예) # head /etc/systemd/user.conf

- **more**
  텍스트로 작성된 파일을 화면에 페이지 단위로 출력
  예) # more /etc/systemd/system.conf

- **less**
  more와 용도가 비슷하지만 기능이 더 확장된 명령
  예) # less /etc/systemd/system.conf

- **file**
  File이 어떤 종류의 파일인지를 표시
  예) # file /etc/systemd/system.conf

- **clear**
  터미널 화면을 깨끗하게 지워줌
  예) # clear

- **nano**
  텍스트 편집기

리눅스 CLI 실습

일반 파일 이동

1. temporary의 파일 확인
2. mv 명령어로 say-hi 파일의 이름 변경
3. greetings 파일의 이름 변경
4. 파일 경로 변경
5. temporary 디렉터리 조회
6. hello 파일을 현재 작업 디렉터리로 이동
7. 파일 이름 변경

```
파일 경로 변경

# `say-hello` 파일을 다른 디렉터리로 이동 (예: ~/Documents)
mv say-hello ~/Documents

temporary 디렉터리 조회

# 다시 temporary 디렉터리로 이동한 후 목록 확인
cd ~/temporary
ls

hello 파일을 현재 작업 디렉터리로 이동

# hello 파일을 현재 디렉터리로 이동 (예: ~/temporary에서 다른 위치로 이동)
mv ~/temporary/hello .  # 현재 디렉터리로 이동

hello 파일의 이름 변경

# hello 파일의 이름을 hi로 변경
mv hello hi
```

```
temporary 디렉터리의 파일 확인

# temporary 디렉터리로 이동
cd ~/temporary

# 현재 디렉터리의 파일 목록 확인
ls

say-hi 파일의 이름 변경

# say-hi 파일의 이름을 say-hello로 변경
mv say-hi say-hello

greetings 파일의 이름 변경

# greetings 파일의 이름을 welcome으로 변경
mv greetings welcome
```

디렉터리 이동

1. 현재 디렉터리 확인
2. haha 디렉터리 생성
3. hoho로 이름 변경
4. 디렉터리 위치 이동
5. 디렉터리 이동하며 hihi로 이름 변경

현재 디렉터리 확인

```
# 현재 작업 중인 디렉터리 확인
pwd
```

haha 디렉터리 생성

```
# 현재 디렉터리에 `haha`라는 새 디렉터리 생성
mkdir haha
```

haha 디렉터리 이름을 hoho로 변경

```
# `haha` 디렉터리의 이름을 `hoho`로 변경
mv haha hoho
```

hoho 디렉터리로 이동

```
# `hoho` 디렉터리로 이동
cd hoho
```

디렉터리 이동과 동시에 hihi로 이름 변경

```
# `hoho` 디렉터리를 현재 경로에서 상위 경로로 이동시키면서 `hihi`로 이름 변경
mv ../hoho ../hihi

# 상위 디렉터리로 이동하여 변경된 디렉터리 확인
cd ..
ls
```

일반 파일 삭제

1. 현재 상태 확인
2. say-hi 파일 삭제
3. -i 옵션을 주고 hello 파일 삭제
4. 여러 파일 한꺼번에 삭제

현재 상태 확인

```
# 현재 디렉터리에 있는 파일 목록을 확인
ls
```

say-hi 파일 삭제

```
# `say-hi` 파일을 삭제
rm say-hi
```

-i 옵션을 주고 hello 파일 삭제

```
# `hello` 파일을 삭제할 때 확인 메시지 표시
rm -i hello
# 삭제 여부를 묻는 메시지가 나오면 `y`를 입력해 삭제를 확정
```

여러 파일 한꺼번에 삭제

```
# 예를 들어 `file1.txt`, `file2.txt`, `file3.txt` 파일을 한꺼번에 삭제
rm file1.txt file2.txt file3.txt
```

또는, 특정 패턴에 맞는 파일을 모두 삭제할 수 있습니다. 예를 들어, .txt 확장자를 가진 파일 모두 삭제:

```
rm *.txt
```

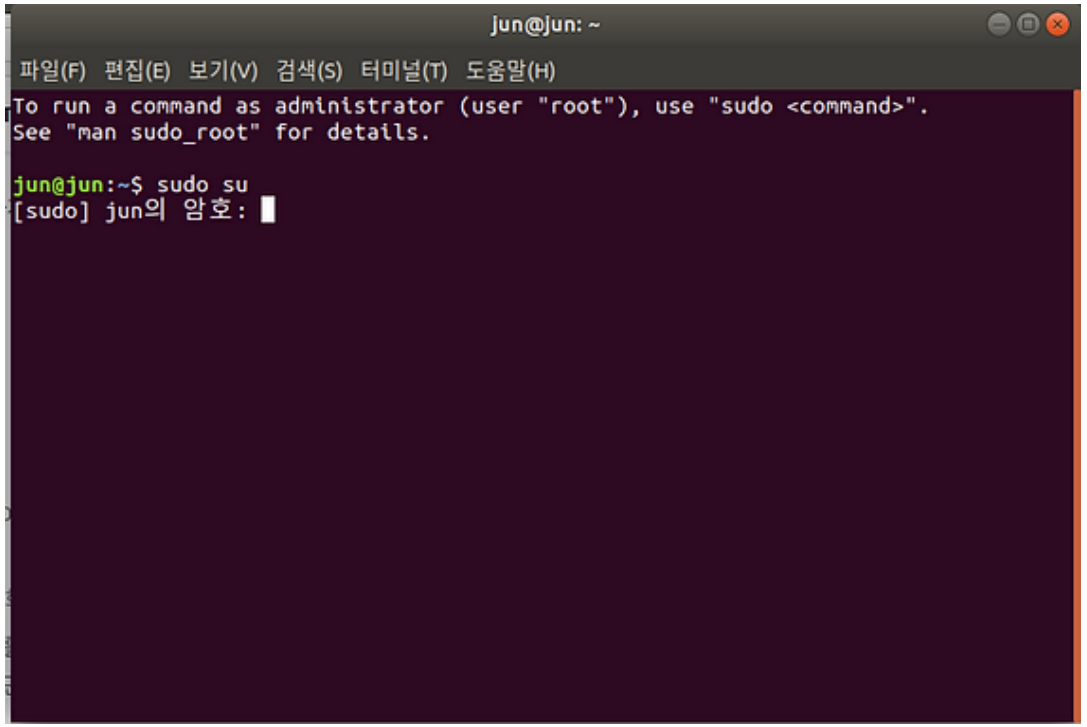
우분투 터미널

우분투 터미널은 사용자가 컴퓨터에 명령어를 입력하여 시스템을 제어 관리할 수 있습니다

아래에는 유용한 터미널 단축키들을 모았습니다

터미널 단축키

Ctrl + C: 현재 작업 중지
Ctrl + Z: 작업 일시 중지
Ctrl + D: 터미널 로그아웃 또는 종료
Tab: 자동 완성
Ctrl + R: 명령어 기록에서 검색



리눅스 Terminator, 커널, 쉘, gedit, bash

터미널

터미널(terminal): 컴퓨터와 사용자 간에 상호작용할 수 있게 연결하는 장치

입력 장치: 사용자가 컴퓨터에 명령을 전달하는 장치

출력 장치: 컴퓨터가 사용자에게 결과를 보여주는 장치

셸(shell)

운영체제가 제공하는 명령어 기반 인터페이스

셸 스크립트(shell script): 셸에서 동작 가능한 명령을 모아놓은 파일

Bash(배시)

리누스 토르발즈가 리눅스를 개발할 때 리눅스로 처음 포팅한 프로그램

Gedit

초보자 친화적인 에디터

```
apt-get install gedit -y
```

터미널과 셸의 관계

터미널은 사용자와 컴퓨터가 상호작용하기 위한 매체

이때 사용하는 도구 중 하나가 바로 셸

컴퓨터가 부팅되면 운영체제는 터미널을 통해 사용자에게 내용을 보여주거나

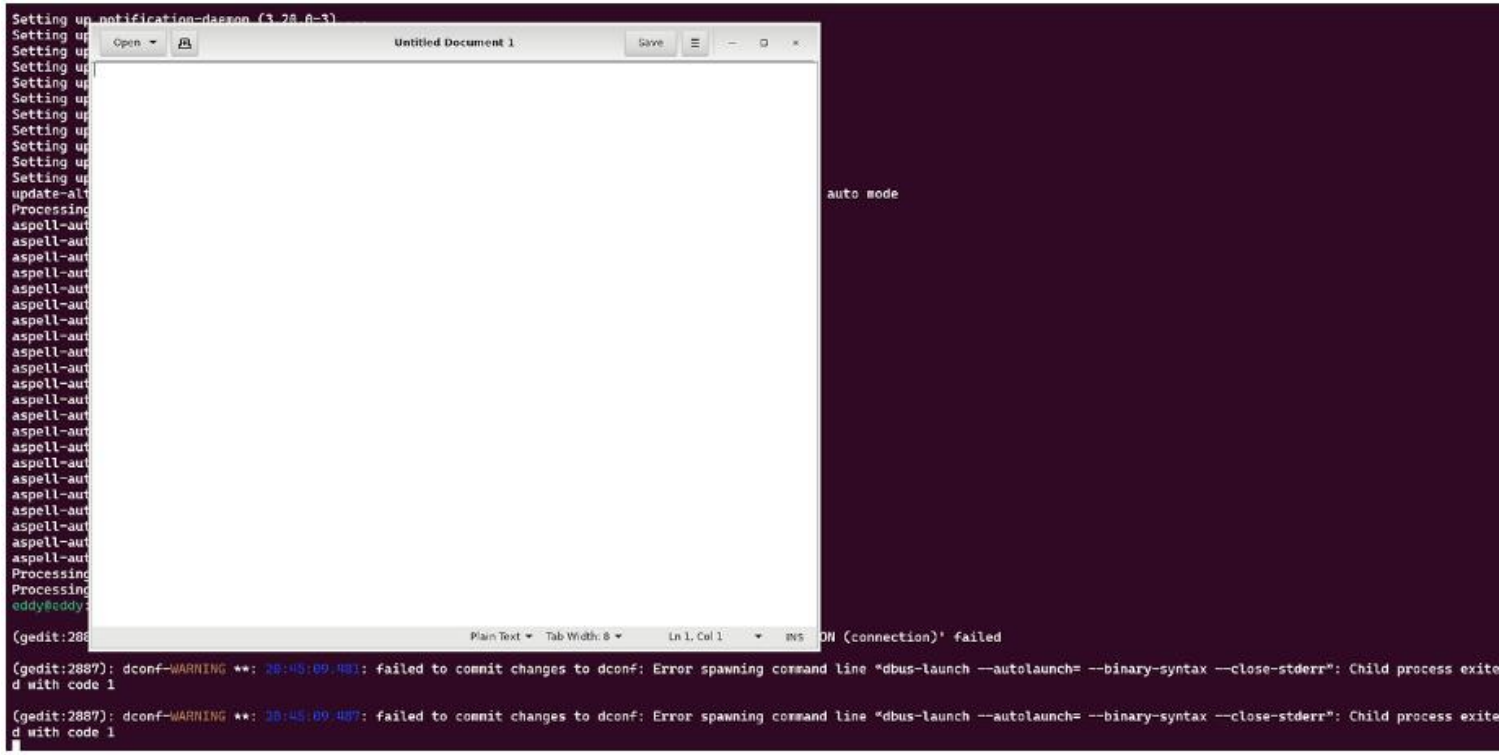
사용자로부터 명령을 입력 받을 수 있는 상태가 됨

gedit 실습

초보자 친화적인 에디터

```
apt-get install gedit -y
```

```
gedit
```



ROKEY BOOT CAMP

수고하셨습니다.