FUNDAMENTALS\_SO3

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Ready to go

1-1. LMS 200% 활용하기

1-2. 안녕, 우분투

1-3. 삐빅! 환경 구성 나와라, 오버

1-4. 커널 실행하기

1-5. 코드 실행 확인하기

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

2. 인공지능 개발자가 되기 위한 위대한 첫 걸음

2-1. 인공지능 개발자가 되기 위한 위대한 첫 걸음

AIFFEL 온라인 LMS에서는 학생 여러분의 편의를 위해 클라우드 컨테이너 를 제공하고 있습니다

개인 PC(Ubuntu 20.04.3, Python 3.9.7, TensorFlow 2.6.0 설치)

공식 텐서플로우 문서 Effective TensorFlow 2 | TensorFlow Core https://www.tensorflow.org/guide/effective\_tf2

2-2. 까만 화면에 흰 글씨, 해커만 쓰는 거 아니냐고? 나도 쓸 수 있다!

터미널(Terminal) 터미널 환경을 우리는 CLI(Command Line Interface) 라고 합니다.

IDE(Integrated Development Environment) 통합 개발 환경 12 Best Python IDEs and Code Editors in 2022 https://www.softwaretestinghelp.com/python-ide-code-editors/

2-3. 터미널을 자유자재로, 내 친구 삼기

터미널 위에 쉘이 실행되는 것입니다. 터미널은 명령을 입력하는 쉘을 실행하기 위한 토대입니다. 터미널이 TV라면 쉘은 방송입니다.

TV가 켜짐과 동시에 방송이 나오는 것처럼 터미널이 실행됨과 동시에 쉘이 실행되는 것이죠

터미널, 콘솔, 쉘의 차이점 https://blog.naver.com/PostView.naver?blogId=asianchairshot&logNo=221383363419&redirect=Dlog&widgetTypeCall=true&topReferer=https%3A%2F%2Flms.aiffel.io%2F&directAccess=false

whoami, pwd, ls, cd, mkdir, rm

리눅스 ls -l 명령어 - 파일 정보 확인과 의미 https://www.leafcats.com/137

2-4. Terminal, 또 무엇을 할 수 있을까?

패키지(Package) 란, 간단히 말해서 특정 기능을 하는 작은 프로그램 단위

sudo apt list --installed 지금까지 설치된 패키지의 리스트를 한 번 확인

sudo apt list --installed 명령어 뒤에 | grep (패키지명) 을 입력해주면 검색하는 단어를 포함하는 패키지만 출력

Linux의 apt-get 명령어 https://codechacha.com/ko/linux-apt-commands/

apt-get update, apt-get upgrade, apt-get install, apt-get remove, apt-get purge, apt-get autoremove

2-5. 가상환경으로 패키지를 조금 더 유연하게 관리해보자

가상환경(virtual environment) 컴퓨터에 설치된 패키지 간의 충돌 또는 패키지 버전에 의한 이슈 등을 방지하기 위해 가상으로 나누어서 사용하는 환경

즉, 특정 프로그램을 돌리기 위해 필요한 패키지들을 모아 만든 각각의 독립된 방과 같은 개념

파이썬 가상환경을 관리하는 툴은 pyenv, pip, anaconda 등 다양합니다

--------------- conda -----------------

Ubuntu 20.04 - Anaconda 설치 및 가상환경 구축 https://mishuni.tistory.com/118

Create virtual environments for python with conda https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/

가상환경 생성 및 실행

(1) 아나콘다 설치

Ubuntu 20.04 - Anaconda 설치 및 가상환경 구축

$ which conda /opt/conda/bin/conda

여기서 which 명령어는 프로그램 설치 경로를 확인하는 것입니다.

윈도우즈에서는 where를 사용합니다.

프로그램 설치 여부와 경로를 확인하는데 유용하니 알아두세요.

아래 명령어로 설치된 아나콘다 버젼을 확인할 수 있습니다.

$ conda --version

conda 4.10.3

(2) 가상환경 생성 및 실행

이제 가상환경을 직접 만들어 봅시다.

$ conda create -n my\_env\_name python=3.9.7

-n 뒤에 my\_env\_name 과 같이 원하는 가상환경의 이름을 입력하면 됩니다.

python=3.9.7 은 이 가상환경에서는 3.9.7 버전의 python을 사용하겠다는 의미입니다.

가상환경이 잘 만들어졌는지 가상환경 리스트를 확인해봅시다.

$ conda env list

# conda environments:

#

base \* /opt/conda

my\_env\_name /opt/conda/envs/my\_env\_name

가상환경 리스트에 방금 만든 환경이 나타났다면 성공입니다!

\* 는 현재 가상환경으로 어떤 환경을 사용하고 있는지 나타냅니다.

가상환경을 사용하려면 만든 후에 activate 을 해줘야 합니다.

가상환경을 설명할 때 "독립된 공간"이라고 설명했던 내용 기억나시나요?

activate 은 문을 열고 그 공간에 들어가는 개념으로 생각하면 됩니다.

$ conda activate my\_env\_name

CommandNotFoundError: Your shell has not been properly configured to use 'conda activate'.

To initialize your shell, run

$ conda init <SHELL\_NAME>

Currently supported shells are:

- bash

- fish

- tcsh

- xonsh

- zsh

- powershell

See 'conda init --help' for more information and options.

IMPORTANT: You may need to close and restart your shell after running 'conda init'.

클라우드에서는 위와같은 에러가 뜰 테지만 당황하지 마시고 conda init 명령어를 실행후 터미널을 재실행 합니다!

$ conda init

이제 터미널을 재실행 해주세요!

$ conda activate my\_env\_name

(my\_env\_name) root@wobqrt210kek94cki3g4i8hr5-6bfb9c97d9-6zfxd:~#

$ conda env list

# conda environments:

#

base /opt/conda

my\_env\_name \* /opt/conda/envs/my\_env\_name

다시 리스트를 확인해보면 우리의 가상환경이 활성화 된 것을 확인할 수 있습니다.

이제 새로운 가상환경에 필요한 라이브러리들을 이것저것 설치해서 이 공간을 꾸밀 수 있습니다!

먼저 이 가상환경 내에 기존에 설치되어 있는 라이브러리 리스트를 확인해볼까요?

$ pip list

Package Version

---------- -------------------

pip 21.3.1

setuptools 58.4.0

wheel 0.37.0

여러 가지 라이브러리가 이미 설치되어 있네요.

이 가상환경 내에 텐서플로우(TensorFlow)도 설치되어 있을까요? 검색해봅시다.

$ pip list | grep tensorflow

위에서 사용해봤던 것처럼, grep 명령어를 사용하면 원하는 키워드를 검색할 수 있습니다.

그렇다면 새로운 가상환경에 텐서플로우를 설치해봅시다.

텐서플로우를 원하는 버전으로 설치하고 싶다면 다음과 같이 버전을 지정해 입력해주면 됩니다.

$ pip install tensorflow==2.6.0

(참고) GPU 버전의 텐서플로우를 설치하려면 pip install tensorflow-gpu로 지정해 주어야 합니다.

모두 설치한 후 다시 리스트를 확인해봅시다.

$ pip list | grep tensorflow

tensorflow 2.6.0

리스트에서 tensorflow가 나타난다면 정상 설치 된 것입니다.

이렇게 각각의 가상환경을 만들어서 패키지의 종류와 버전을 관리할 수 있죠!

예를 들어 여러분들이 AIFFEL에서는 텐서플로우2를, 개인 프로젝트에서는 텐서플로우1을 사용하고 싶으시다면, 서로 다른 가상환경을 만들어 관리하면 됩니다.

하나의 가상환경을 종료하고 싶다면, deactivate 명령어를 사용하면 됩니다.

문을 닫고 바깥 공간으로 나오는 것이라고 생각하시면 됩니다.

$ conda deactivate

가상환경을 잘못 만들었다면 다음 명령어로 삭제할 수 있습니다.

$ conda env remove -n my\_env\_name

이 외에도 아나콘다를 다루는 다양한 명령어는 다음과 같은 cheatsheet에서 확인할 수 있습니다.

Conda Cheat Sheet https://docs.conda.io/projects/conda/en/4.6.0/\_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

개인 공간에 저장해두고 필요할 때 사용하는 것도 좋은 방법입니다!

2-6. 마무리

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3-1. 터미널로 배우는 리눅스 운영체제 | 5분

3-2. 떠나자 디지털 월드로! (1) pwd | 10분

echo: 뒤이어 입력한 내용을 출력

pwd: Print Working Directory의 약자(또는 Present Working Directory의 약자)입니다.

현재 내가 위치한 디렉토리 위치를 출력

3-3. 떠나자 디지털 월드로! (2) ls | 10분

ls: 현재 디렉토리 내의 파일 및 폴더들을 출력

ls --all 모두 표시 --all은 -a로 줄여서 쓸 수도 있다

ls --help를 입력해 봅시다

-l: 각 파일 및 폴더의 권한, 소유자, 크기, 수정일시 등 자세한 정보를 출력합니다.

-h: 파일 크기를 바이트 수 대신 사람이 알아볼 수 있는 단위(1K, 234M, 2G 등)로 표시합니다.

-t: 파일 수정시간 순으로 정렬해서 출력합니다. 최신 파일이 위쪽으로 표시됩니다.

-r: 정렬 순서를 뒤집어 출력합니다. -rt의 형태로 쓰면 최신 파일이 아래쪽으로 표시되어 유용할 때가 있습니다.

한 글자 인자를 여럿 쓸 때에는 서로 붙여 쓸 수 있습니다. 즉, ls -l -a -h는 ls -lah로 쓸 수 있습니다.

Tip) ls -alrt 가 유용할 때가 많습니다

Ctrl+Shift+↑/↓으로 한줄씩 위/아래로 스크롤하거나,

Shift+PageUp/PageDown으로 한 페이지씩 위/아래로 스크롤 할 수 있습니다

3-4. 떠나자 디지털 월드로! (3) cd | 10분

cd: Change Directory의 약자로, 뒤에 따라오는 폴더로 내가 있는 셸의 현재 위치를 이동

cd .. 상위 폴더로 이동

여기서 ..나 aiffel에서처럼 명령을 입력하는 시점에 내가 현재 있는 디렉토리에 영향을 받는 위치 표시 방식을 상대 경로라고 합니다.

반면에 /home/june/aiffel와 같이 어디서 입력하든 똑같은 곳을 가리키도록 위치 전체를 적어주는 것을 절대 경로

/home: 사용자별 홈 디렉토리들이 있는 곳

/root: 최고 관리자 계정의 홈 디렉토리

/mnt: 저장장치(HDD, SSD)가 붙는 위치

/media: 이동식 미디어(USB 드라이브)가 붙는 위치

/tmp: 재부팅 시 삭제될 임시 파일들을 저장하는 폴더

/dev: 컴퓨터에 연결된 하드웨어 및 가상 기기(device)들을 가리키는 파일들

/proc: 현재 실행 중인 프로세스들을 가리키는 파일들

/etc: 각종 설정 파일들

/bin: 실행 가능한 프로그램(binary)들

/sbin: 시스템 관리용 프로그램들

/usr: 다중 사용자 모드에서 사용 가능한 파일 및 프로그램들 (root 계정만 있는 단일 사용자 모드에서는 사용 불가)

/var: 캐시, 로그 등 시스템 구동 간 계속 내용이 바뀌는 파일들

3-5. 운영체제 일반 개론 (1) 운영체제란 | 5분

운영체제란 컴퓨터의 다양한 하드웨어 장치들과 더불어 소프트웨어를 관리하는 시스템

windows 8 = 운영체제의 버전이 8, Version 6.3 (Build 9600) = 커널의 버전이 6.3이라는 뜻

3-6. 운영체제 일반 개론 (2) 커널 | 5분

운영체제의 내부는 알맹이인 커널(kernel)과 껍데기인 셸(shell)로 이루어져 있습니다

체 운영체제의 버전이 8이고, 커널의 버전이 6.3이라는 뜻

3-7. 운영체제 일반 개론 (3) 셸 | 5분

셸은 말 그대로 눈에 보이는 운영체제의 껍데기인데,

그래픽 기반의 GUI(Graphical User Interface) 셸과

텍스트 기반의 CLI(Command Line Interface) 셸로 구분

3-8. 운영체제 일반 개론 (4) 프로세스와 스레드 | 5분

프로그램이 실제로 작업을 수행될 때 운영체제는 이를 프로세스(process) 라는 단위로 관리합니다.

운영체제는 프로세스 단위로 메모리와 CPU 연산 등 컴퓨팅 자원을 분배합니다.

메모리 관리 : 운영체제는 프로세스가 컴퓨터의 전체 메모리 어디든지 쓸 수 있도록 하는 것이 아니라

가상 메모리(virtual memory)로서 일부만 떼어서 제공해줌으로써,

다른 프로세스의 메모리 또는 운영체제 자체가 사용하고 있는 커널 메모리를 훔쳐볼 수 없도록 합니다.

이것을 운영체제의 중요한 보안 기제 중 하나인 프로세스 격리(process isolation)라고 부르며,

때문에 프로세스들이 서로 소통하기 위해서는 별도로 허용된

프로세스 간 통신 (Inter-Process Communication, IPC) 기법들을 사용해야 합니다.

CPU 자원 관리 : 메모리는 주소로서 프로세스에게 제공되지만, CPU 연산의 경우 코어 단위로 프로세스에 제공됩니다.

프로세스는 스레드(thread) 라는 단위로 코어를 하나씩 사용할 수가 있는데,

프로세스가 하나 시작되면 기본적으로 하나의 스레드를 가지고 시작하지만,

개발자는 프로그램이 더 많은 스레드를 사용하도록 설계하여 다중 CPU 코어의 이점을 극대화할 수 있습니다.

프로세스는 운영체제가 메모리를 할당하는 작업단위라면,

스레드는 프로세스가 할당받은 메모리를 활용하는 실행단위입니다.

하나의 프로그램 안에서 여러개의 프로세스를 활용하는 것을 멀티프로세싱(Multiprocessing) ,

하나의 프로세스 안에서 여러개의 스레드를 활용하는 것을 멀티스레딩(Multithreading) 이라고 합니다

3-9. 리눅스 기반 운영체제 (1) 터미널 | 10분

우분투에서 Cloud shell 열기면 터미널 또는 콘솔이라고도 부르는 창(CLI 셸)이 열립니다.

이 창을 여는 프로그램은 엄밀히 말하면 터미널이 아니라,

정확히는 그래픽 환경에서 터미널을 모사하는 터미널 에뮬레이터입니다.

터미널이라는 이름은 개인용 컴퓨터(PC)가 등장하기 전의 시절,

실제 연산을 담당하는 크고 비싼 메인프레임 컴퓨터와는 별도로,

입출력만 담당하는 CRT 모니터와 키보드로 구성된 말단(=terminal)의 기기에서 유래한 명칭입니다.

여러 대의 터미널을 한 대의 메인프레임에 연결해 여러 사용자끼리 나눠 사용하는 것도 가능했으며,

원격으로 타이핑을 한다는 의미에서 teletype이라는 명칭으로도 불렸습니다.

시대가 흘렀지만 teletype은 운영체제 안에 흔적이 남아있습니다.

/dev(device, 장치) 폴더 안에는 tty(teletype)로 시작하는 여러 대의 소프트웨어 터미널이 있습니다.

물론 시리얼 포트를 통해 실제 하드웨어 터미널과 연결할 수 있는 ttyS도 함께 남아있습니다.

$ ls /dev

core full null nvidiactl nvidia-uvm-tools pts shm stdin termination-log urandom

fd mqueue nvidia0 nvidia-uvm ptmx random stderr stdout tty zero

우분투에서는 18.04부터 로그인한 사용자의 그래픽 환경을 2번 터미널에 붙여두기 때문에,

우리는 CLI 셸(tty3)에서 무사히 GUI 셸(tty2)으로 귀환할 수 있었습니다.

즉, 터미널이란 컴퓨터에 정보를 입력하고 출력하는 소프트웨어 및 하드웨어로써의 장치이며,

지금도 우리는 하나의 터미널(tty2)이라는 창구를 통해 컴퓨터 안의 세상을 그래픽으로써 들여다보고 있는 것입니다.

아래 영상의 장치도 터미널이라 부르기에 충분합니다. https://youtu.be/2XLZ4Z8LpEE

3-10. 리눅스 기반 운영체제 (2) 프로세스 관리와 시스템 콜 | 10분

주피터 노트북이 돌아가고 있다면 새로운 탭 또는 창으로 새로운 터미널 세션을 시작해 주세요.

👍️ ps: 현재 터미널과 관련된 프로세스 목록을 출력합니다.

👍️ ps -ef: 모든 터미널에서 현재 실행중인 프로세스 목록 및 부가정보를 출력합니다.

$ ps -ef

UID PID PPI D C STIME TTY TIME CMD

root 1 0 0 05:10 ? 00:00:00 /bin/sh -c jupyter notebook --NotebookApp.token=$DEFAULT\_JU

root 6 1 0 05:10 ? 00:00:07 /opt/conda/bin/python /opt/conda/bin/jupyter-notebook --Note

root 7 1 0 05:10 ? 00:00:00 ttyd -p 8887 /bin/bash

root 10 7 0 05:50 pts/0 00:00:00 /bin/bash

root 13 7 0 05:53 pts/1 00:00:00 /bin/bash

root 17 7 0 05:55 pts/2 00:00:00 /bin/bash

root 20 7 0 06:02 pts/3 00:00:00 /bin/bash

root 66 7 0 07:23 pts/4 00:00:00 /bin/bash

root 69 66 0 07:47 pts/4 00:00:00 ps -ef

PID는 process ID 번호이고, TTY는 해당 프로세스가 붙어있는 teletype(터미널)입니다.

0번 가상 터미널(pts, pseudo teletype slave)에 우리가 방금 실행한 ps 명령어와, 해당 명령어를 실행한 셸 bash가 떠있습니다.

TTY 칸에 ?가 표시된 프로세스들은 터미널에 부착되지 않은 프로세스들입니다.

이제 한번 프로세스를 하나 죽여봅시다. GUI 셸의 핵심인 Xorg가 만만해 보입니다.

물론 실제로 망가뜨리진 않겠지만, 열려있던 창이 모두 종료될 테니 작업 중이던 내용을 모두 저장한 다음,

위에서 확인한 /usr/bin/Xorg 프로세스의 번호를 넣어 sudo kill -s KILL [PPID]를 입력합니다.

[PPID] 부분에 여러분의 컴퓨터에서 실제로 출력한 숫자를 집어넣으세요.

여러분의 터미널에서 나오는 프로세스 번호는 GUI가 켜질 때마다 다르게 부여될 수 있으니 아래와 같이 꼭 직접 확인하세요.

$ ps -ef | grep Xorg

root 71 66 0 07:54 pts/4 00:00:00 grep Xorg

👍️ grep: 입력으로 전달된 내용에서 특정 문자열을 포함한 라인만 선택해서 출력. 파이프(|)를 통해 다른 명령문 실행결과와 결합해서 사용할 때 특히 유용

❤️ sudo: 이어지는 명령을 최고 관리자 권한으로 실행합니다.

👍️ kill: PID에 해당하는 프로세스에 시그널을 보냅니다.

$ sudo kill -s KILL 66

Xorg가 죽으면서 화면이 잠시 꺼졌다가, 자동으로 재시작하면서 다시 로그인 화면으로 돌아올 겁니다.

이런 식으로 3번 터미널의 CLI로 전환해서 GUI를 죽이고

다시 2번 터미널의 GUI로 돌아오는 방법으로 재부팅 없이 해결할 수 있습니다.

물론 GUI 상에서 돌아가고 있던 프로그램들의 정보는 날아갈 수 있으니 주의해야 합니다.

그럼 Xorg 프로세스의 종료는 어떤 방식으로 이뤄진 걸까요?

프로세스 관리는 운영체제의 고유 권한이기 때문에, 우리는 커널의 도움이 필요합니다.

우리가 CLI 셸에 kill 명령어를 입력하는 순간, 셸의 프로세스는 커널에게 이 명령을 전달하고, 커널은 권한 등을 확인한 후에 이를 수행합니다.

이렇게 프로세스가 커널에게 요청을 전달하는 기능을 시스템 콜(system call) 이라고 합니다.

시스템 콜은 운영체제의 커널이 프로그램에 제공하는 인터페이스이며,

프로세스가 직접 할 수 없는 프로세스 제어(kill 등), 파일 조작(폴더 생성 mkdir, 파일 읽기 read 등),

장치 및 자원 관리, 네트워크 통신 등을 가능케 합니다.

반대로 보면, 각 프로세스는 커널을 통하지 않고서는 파일이나 장치, 또는 다른 프로세스에 간섭할 수 없다는 뜻이 됩니다.

참고로 kill은 그 자체로 프로세스를 죽이는 명령이 아니라, 프로세스에 신호를 보내는 명령어입니다.

강제종료하는 KILL 외에도, 프로그램 실행 중에 Ctrl+C를 누르면 전달되는 INT (interrupt) 신호나,

프로세스를 종료하기 전에 해당 프로세스에게 해명할 기회를 주는 TERM (terminate) 신호도 있습니다.

3-11. 리눅스 기반 운영체제 (3) 프로그램과 환경변수 | 10분

명령어들의 진짜 정체는 두 가지로 구분할 수 있습니다.

1) 실제 프로그램과

2) 셸에서 제공하는 셸 명령어입니다. which 명령어를 사용해서 그 정체를 한번 밝혀봅시다.

👌️ which: 명령어의 전체 경로를 출력합니다.

$ which ls

/bin/ls

$ which cd

짜잔! ls는 /bin 디렉토리에 설치된 ls라는 프로그램이고,

cd는 프로그램이 아니라 셸에서 제공하는 명령어였기에 아무 결과도 출력되지 않습니다.

우리가 사용하는 셸은 bash였죠? 셸도 프로그램일 테니 한번 경로를 확인해 봅시다.

$ which bash

/bin/bash

즉, 우리가 셸에 명령어를 입력하면 셸은 일단 셸 내에서 정의된 명령인지 확인합니다.

만약 셸에 정의된 명령 중 있다면, 그대로 명령을 실행합니다.

셸에 정의된 명령이 아니라면, 설치된 프로그램 중에서 해당 이름을 가진 프로그램이 있는지 확인하여 실행합니다.

그럼 해당 이름을 가진 프로그램이 실제로 설치되어 있는지는 어디서 확인할까요? echo $PATH라고 입력하여 한번 물어봅시다.

$ echo $PATH

/opt/conda/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

무언가 나왔습니다. 일단 우리가 입력한 명령을 먼저 이해해 보겠습니다.

echo는 뒤에 입력된 내용을 출력하라는 명령어였는데, 뒤의 $PATH는 무엇일까요?

셸에서 단어 앞에 $가 붙으면 환경 변수(environment variable) 를 의미합니다.

파이썬의 변수처럼, 환경 변수는 운영체제가 프로세스 단위로 사용하는 변수입니다.

즉, 우리가 echo $PATH는 PATH라는 이름의 환경 변수($)의 값을 출력(echo)하라는 뜻입니다.

PATH 환경 변수는 셸이 프로그램을 실행할 때 참조할 경로들을 나타냅니다.

값을 보면 디렉토리 여러 개가 :로 구분되어 있는 것을 알 수 있는데,

만약 동일한 이름의 프로그램이 여러 디렉토리에 설치되어 있다면,

PATH 내 순서 상 앞에 있는 디렉토리의 프로그램을 실행합니다.

아나콘다(Anaconda)를 설치했다면,

우분투에 기본으로 설치된 파이썬 대신 아나콘다가 설치한 파이썬을 사용하도록 PATH의 값이 변경되어 있을 겁니다.

마찬가지로 which python을 통해 확인해볼 수 있습니다.

참고로 PATH 환경 변수의 기본값은 사용자가 로그인할 때 운영체제가 자동으로 정의하지만,

사용자가 직접 수정할 수도 있습니다.

셸 명령어인 export 명령어를 통해 PATH의 앞에 /aiffel 디렉토리를 추가해 봅시다.

👌️ export: 셸 변수나 함수를 현재 환경으로 내보냅니다.

$ echo $PATH

$ export PATH=/aiffel:$PATH

$ echo $PATH

이렇게 PATH환경변수에 디렉토리 경로를 추가하면 셸이 특정 프로그램을 실행할때 참조할 수 있게 되는 것입니다

또한 env 명령어로 현재 설정된 환경변수들을 모두 출력할 수 있습니다.

👌️ env: 명령어가 제공된다면 해당 명령어를 새로운 환경에서 실행합니다.

명령어가 제공되지 않은 경우 현재 환경의 정보를 출력합니다.

$ env

CUDNN\_VERSION=8.0.5.39

WTETYSOCXVX7E6AVQWHCTCCAH\_PORT\_8887\_TCP\_ADDR=10.0.253.249

LC\_ALL=en\_US.UTF-8

WTETYSOCXVX7E6AVQWHCTCCAH\_PORT\_8888\_TCP\_PROTO=tcp

LD\_LIBRARY\_PATH=/usr/local/nvidia/lib:/usr/local/nvidia/lib64

WTETYSOCXVX7E6AVQWHCTCCAH\_SERVICE\_PORT=8888

LANG=en\_US.UTF-8

HOSTNAME=wtetysocxvx7e6avqwhctccah-776475db6-h2swf

CONDA\_DIR=/opt/conda

CONDA\_VERSION=4.9.2

NVIDIA\_VISIBLE\_DEVICES=GPU-69bafdf7-9469-6565-605e-81ec2a4871ed

KUBERNETES\_PORT\_443\_TCP\_PROTO=tcp

KUBERNETES\_PORT\_443\_TCP\_ADDR=10.0.0.1

NCCL\_VERSION=2.8.4

TF\_FORCE\_GPU\_ALLOW\_GROWTH=true

KUBERNETES\_PORT=tcp://10.0.0.1:443

PWD=/aiffel

HOME=/aiffel

....

우리가 /bin 디렉토리 안의 프로그램 bash를 현재 셸(SHELL)로서 사용한다는 내용도 나와있네요.

3-12. 리눅스 기반 운영체제 (4) 패키지 관리자 | 10분

우분투 운영체제는 공식적으로 APT(Advanced Packaging Tool)라는 패키지 관리자(package manager) 를 제공합니다.

패키지 관리자란 공인 저장소에서 프로그램과 라이브러리(다른 프로그램들이 참조하여 사용할 수 있는 코드)를

패키지 단위로 설치하고 관리할 수 있도록 하는 프로그램입니다.

우분투에는 APT말고도 Snap이라는 비교적 새로운 프로그램도 있으며,

CentOS의 YUM(Yellowdog Updater Modified),

Red Hat 계열의 RPM(Red Hat Package Manager) 등 각 리눅스 계열 운영체제는 다양한 기본 패키지 관리자를 제공합니다.

운영체제 상 패키지 관리자 외에도, 파이썬의 PIP(Python Package Installer) 등 프로그래밍 언어상 제공되는 경우도 있습니다.

APT로 우분투에 htop이라는 프로그램을 설치해 보겠습니다.

htop은 CPU 및 메모리 사용량을 시각화하고, 프로세스 별 CPU 사용량을 표시해 주는 프로그램입니다.

❤️ apt-get: APT 패키지 관리자를 통해 공인 저장소에서 패키지를 가져다 설치하거나 제거합니다.

우선 등록된 공인 저장소에 있는 패키지 목록을 최신 버전으로 가져옵니다.

$ sudo apt-get update

Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]

Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease

Get:3 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [24.7 kB]

Get:4 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1,414 kB]

Get:5 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]

Get:6 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [450 kB]

Get:7 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2,184 kB]

Get:8 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]

Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [480 kB]

Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [2,184 kB]

Get:11 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [33.5 kB]

Get:12 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2,616 kB]

Fetched 9,639 kB in 1s (6,596 kB/s)

Reading package lists... Done

이제 설치를 진행합니다. 패키지 목록을 읽고, 해당 패키지가 의존하는 다른 패키지들이 있는지 등을 자동으로 확인한 뒤, 설치를 진행합니다.

$ sudo apt-get install htop

설치가 완료되면 바로 실행해 봅시다. htop을 종료하려면 q를 누르면 됩니다.

$ htop

삭제는 설치만큼 쉽습니다. apt-get 뒤에 install 대신 remove를 넣어서 프로그램만을 제거하거나,

purge를 넣어 해당 프로그램의 설정 파일까지 모두 삭제시킬 수 있습니다.

만약 중간에 확인([Y/n])을 물으면 y를 누르고 엔터를 쳐서 제거를 승인합니다.

$ sudo apt-get remove htop

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be REMOVED:

htop

0 upgraded, 0 newly installed, 1 to remove and 44 not upgraded.

After this operation, 221 kB disk space will be freed.

Do you want to continue? [Y/n] y

3-13. 리눅스 기반 운영체제 (5) 사용자와 권한 | 10분

sudo 명령어

한 대의 컴퓨터를 여러 사람이 함께 사용하는 것은 운영체제의 초기 역사부터 일반적인 사용법이었습니다.

운영체제에서 어떤 작업을 실행하기 위해서는 어떤 사용자 이름으로 실행할지 미리 정해져 있어야 한다는 뜻입니다.

ps -ef 명령어를 사용하면 프로세스마다 누구의 권한으로 실행되는지(UID, User ID) 확인할 수 있습니다.

프로세스는 현재 사용자가 가진 범위만큼의 권한을 가집니다.

우리가 위에서 cd /root로 최고 관리자의 디렉토리에 접근하려고 했을 때처럼,

해당 사용자의 권한을 넘어서는 작업을 요청하면 운영체제의 커널이 이를 거부합니다.

리눅스 기반 운영체제, 윈도우, 맥 OS 모두 공통적으로, 해당 사용자 계정에게 운영체제에 대한 모든 권한이 주어지지는 않습니다.

보안 및 시스템 안정성을 위해 완전한 권한은 운영체제에 기본으로 설정된 최고 관리자(superuser) 계정만이 가지고 있으며,

윈도우에서는 이 계정을 Administrator이라고 부르고, 유닉스 계열 운영체제에서는 root라는 아이디를 가지고 있습니다.

우리가 직접 생성한 사용자 계정은 일반 사용자 계정입니다.

이 때문에 운영체제 자체에 영향을 주는 일(커널 권한이 필요한 프로그램의 설치나 실행 등)은 최고 관리자 권한을 빌려야 합니다.

윈도우에서 무언가를 설치할 때 가끔 보게 되는 "관리자 권한으로 실행" 경고창이나,

유닉스 계열 운영체제에서 사용하는 sudo(superuser do) 명령어가 바로 이 역할을 하며,

일반 사용자가 잠시 최고 관리자(또는 다른 사용자) 권한으로 행동할 수 있게 해줍니다.

사용자 그룹

사용자들이 여러 명 있을 때, 쉽게 묶어서 관리하는 개념이 바로 그룹(group)입니다.

예를 들어 한 사용자가 sudo 명령어를 실행하기 위해서는, 동일한 이름(sudo)라는 이름의 그룹에 속해있어야 합니다.

유닉스 계열 운영체제에서는 설정 파일이나 장치 모두 파일로 접근 가능해야 한다는 철학이 있기 때문에, 그룹 설정도 텍스트 파일로 저장되어 있습니다.

❤️ cat: 하나 이상의 텍스트 파일을 순서대로 출력합니다.

$ cat /etc/group

root:x:0:

daemon:x:1:

bin:x:2:

sys:x:3:

adm:x:4:

tty:x:5:

disk:x:6:

lp:x:7:

mail:x:8:

news:x:9:

uucp:x:10:

man:x:12:

proxy:x:13:

kmem:x:15:

dialout:x:20:

fax:x:21:

접근 권한

프로그램과 디렉토리를 포함한 모든 "파일"에는 소유 사용자와 소유 그룹이 있습니다.

소유 사용자가 할 수 있는 일, 소유 그룹에 소속된 사용자들이 할 수 있는 일,

그리고 그 밖의 사용자들이 할 수 있는 일을 파일마다 설정할 수 있습니다.

이 설정값은 ls -lah 명령을 통해서도 확인할 수 있습니다.

$ ls -lah

total 48K

drwxr-xr-x 1 root root 4.0K Jun 7 08:34 .

drwxr-xr-x 1 root root 4.0K Jun 7 05:10 ..

drwxrwxrwx 2 root root 0 Jun 2 02:45 aiffel

-rw------- 1 root root 12K Jun 7 08:24 .bashrc.swp

drwxr-xr-x 1 root root 4.0K Apr 19 09:30 .cache

drwxrwsr-x 2 root root 4.0K Apr 19 09:30 .conda

drwx------ 3 root root 4.0K Jun 7 08:34 .config

drwxr-xr-x 2 root root 4.0K Jun 4 03:42 data

drwx------ 1 root root 4.0K Apr 19 09:23 .jupyter

drwxr-xr-x 3 root root 4.0K Jun 7 05:10 .local

-rw-r--r-- 1 root root 170 Apr 19 09:28 .wget-hsts

root 가 여러번 나오네요 3번 열은 소유 사용자이고, 4번째 열은 소유 그룹입니다.

위에 두 줄을 보면 현재 디렉토리(/aiffel, 여기서는 .)는 사용자로서 root과 그룹으로서 root이 소유하고 있으며,

그 위의 디렉토리(/, 여기서는 ..)는 사용자와 그룹으로서의 root가 소유하고 있음을 알 수 있습니다.

한번 이걸 변경해볼까요? 현재 디렉토리의 소유 사용자를 root로, 소유 그룹은 adm으로 변경했다가 복구해 보겠습니다.

👍️ chown: 대상 파일의 소유 사용자와 그룹을 변경합니다.

$ sudo chown root:adm .

$ ls -lah

total 48K

drwxr-xr-x 1 root adm 4.0K Jun 7 08:34 .

drwxr-xr-x 1 root root 4.0K Jun 7 05:10 ..

[..생략..]

root@wtetysocxvx7e6avqwhctccah-776475db6-h2swf:/# sudo chown root:root .

root@wtetysocxvx7e6avqwhctccah-776475db6-h2swf:/# ls -lah

total 92K

drwxr-xr-x 1 root root 4.0K Jun 7 05:10 .

drwxr-xr-x 1 root root 4.0K Jun 7 05:10 ..

참고로 위 명령어만으로는 디렉토리 내용물까지 소유자가 변경되지는 않습니다.

디렉토리 내의 모든 파일들에 대해 적용하려면 -R 옵션을 사용해야 합니다.

소유자 외에도, 위 ls -lah의

첫 열을 보면 drwxr-xr-x라고 무언가 적혀있는데, 이 부분이 바로 권한을 표기하는 부분입니다.

첫 글자인 d는 해당 파일이 디렉토리인지 또는 일반 파일인지 여부를 나타냅니다.

그 뒤를 잇는 글자들은 각각 3 글자씩 소유 사용자 , 소유 그룹 , 기타 사용자 에게 허용된

읽기(read, r) , 쓰기(write, w) , 실행하기(execute, x)

예를 들어 drwxr-xr-x는 해당 파일이 디렉토리이고(d),

소유자는 읽고(r) 쓰고(w) 실행(x)하는 모든 동작이 가능하지만,

소유 그룹 소속 사용자와 기타 사용자들은 읽고 실행하기만 가능하다는 것을 표시합니다.

권한은 숫자로도 표시할 수 있습니다.

기본적으로 세 가지 권한 rwx를 각각의 이진수 자리로 봅니다.

세 자리 이진수 숫자를 해석하면 사용자, 그룹, 기타 사용자에게 부여된 권한을 1자리 십진수 숫자로 나타낼 수 있습니다.

예를 들어 rwxr-xr-x를 이진수로 나타내면 111101101이고, 이는 구분 별로 다음과 같습니다.

사용자: 이진수 111 = 십진수 7

그룹: 이진수 101 = 십진수 5

기타 사용자: 이진수 101 = 십진수 5

종합: 755

참고로 실행하기(x)의 경우 프로그램이라면 실제 실행할 수 있다는 것을 나타내지만,

디렉토리의 경우 해당 폴더 안으로 cd로 이동할 수 있는지 여부를 나타냅니다.

chmod 명령어

chmod 명령어를 사용하면 권한을 변경할 수 있습니다. aiffel 폴더의 권한을 한번 변경해 봅시다.

모두에게 모든 권한을 끄기 위해서 숫자 표기로 000을 입력합니다. (000 = 000000000 = ---------)

아래와 같이 aiffel\_test 디렉토리를 만들고 chmod 명령어를 활용하여 권한을 변경하여 보겠습니다. ^^

👍️ chmod: 대상 파일의 권한을 변경합니다.

$ mkdir aiffel\_test

$ ls -lah

total 56K

drwxr-xr-x 1 root root 4.0K Jun 7 09:27 .

drwxr-xr-x 1 root root 4.0K Jun 7 05:10 ..

drwxrwxrwx 2 root root 0 Jun 2 02:45 aiffel

drwxr-xr-x 2 root root 4.0K Jun 7 09:27 aiffel\_test

root@wtetysocxvx7e6avqwhctccah-776475db6-h2swf:~# chmod 000 aiffel\_test/

root@wtetysocxvx7e6avqwhctccah-776475db6-h2swf:~# ls -l

d--------- 2 root root 4096 Jun 7 09:27 aiffel\_test

3-14. 기타 (1) 약간의 하드웨어 1: 연산장치 | 10분

지금까지 CPU, 메모리가 언급되었지만, 당사자인 하드웨어는 제대로 만나보지 못했습니다. 얼굴을 익히기 위해 한번 컴퓨터의 속을 까봅시다.

content img

[https://en.wikipedia.org/wiki/Motherboard]

마더보드의 CPU Socket에는 CPU(Central Processing Unit, 중앙처리장치)가 올라갑니다. 실제로 CPU 칩의 크기는 얇지만, 보통 그 위에 냉각팬을 올리기 때문에 하드웨어를 직접 보면 커보입니다. 이 얇은 칩은 반도체인 실리콘 웨이퍼 위에 집적회로(Integrated Circuits, IC)를 포토리소그래피(photolithography) 공정을 통해 회로를 인쇄해 만듭니다.

content img

[https://en.wikipedia.org/wiki/Wafer\_(electronics)]

칩이 만들어지는 과정이 더 궁금하신 분은 다음 영상을 참고하시면 좋습니다.

기계어

이렇게 제작된 CPU 칩이 처리할 수 있는 명령어는 기계어 라고도 하며, 우리가 사용하는 프로그래밍 언어와는 많이 다릅니다. 기계어로 프로그램을 직접 짜기에는 워낙 복잡하고 비효율적이기에 C, C++, Python 등 인간이 이해할 수 있는 프로그래밍 언어들이 만들어졌습니다. 이렇게 인간이 이해할 수 있는 언어로 짠 프로그램을 CPU에서 실행할 수 있는 기계어로 번역하고 최적화하는 과정이 바로 컴파일 입니다.

명령어 집합

회로 설계는 Intel, AMD 등 제조사마다 다르지만, 지원하는 연산 세트를 나타내는 명령어 집합(instruction set) 은 데스크탑용 CPU 간 상당 부분 호환됩니다. 다만 스마트폰 등 모바일 기기에 들어가는 CPU는 이야기가 다른데, 대부분 ARM 사의 RISC(Reduced Instruction Set Computer)라는 구조를 가지고, 지원하는 명령어는 더 적지만 전력을 더 적게 소비하는 방향으로 설계됩니다.

흥미롭게도 ARM 회사는 더 이상 직접 칩을 만들지 않고, 설계만 만들어 라이센스를 판매합니다. 이 설계를 원형으로 삼성전자(Exynos 등), 애플(A시리즈, M시리즈 등), 퀄컴(SnapDragon 등) 등 회사가 자신들의 목적에 맞게 변경하여 각자 칩을 생산합니다.

메모리와의 관계

CPU를 구분할 때에는 32bit와 64bit로 구분하기도 하는데, 이는 CPU가 한 번에 읽고 쓸 수 있는 데이터의 크기를 나타냅니다. 이 크기가 중요한 이유 중 하나는 사용 가능한 메모리 주소 의 범위에 영향을 주기 때문입니다.

이론적으로 32bit 컴퓨터에서는 2의 32승인 약 42억 개 메모리 주소공간만 사용할 수 있습니다. 1개 주소공간이 1 Byte라면 4GB 크기의 메모리 용량만 표기할 수 있기 때문에, 그 이상의 RAM을 사용할 수 없다는 큰 제한사항이 있었습니다.

CPU의 성능과 발전

CPU의 성능을 판단하는 주요 척도 중 하나는 초당 처리 속도의 기준인 클럭(clock)입니다. 하지만 요새 나오는 CPU를 보면 클럭이 4GHz에서 더 이상 발전하기보다는, 동시에 더 많은 스레드를 처리할 수 있도록 코어 수를 늘리는 방향으로 제조되고 있습니다. 이는 처리 속도를 높이기 위해 소비 전력을 높였다가는 물리적으로 냉각시키기가 너무 어려워지고, 그렇다고 전압을 낮춰서 낮은 전력으로 동작시키기에는 전류가 밖으로 누설되고, 그러면 전력과 전압을 모두 적절하게 유지하면서 집적도를 높이기에는 공정이 너무 미세해지면서(1~5 나노미터) 수율도 떨어지고, 현재의 소재들로는 양자 터널링 현상으로 인해 전자가 닫혀있는 게이트를 그냥 뛰어넘어버리게 되는 복잡한 물리적인/경제적인 이유 때문입니다.

다른 연산장치

이러한 한계로 인해 최근에는 CPU 외에도 특정 작업에 특화된 연산장치들이 떠오르고 있습니다.

,

, 그래픽 연산을 위한 GPU(Graphical Processing Unit) 이 그 중 하나입니다. CPU보다 이해할 수 있는 명령어는 훨씬 적지만, 코어 수를 훨씬 늘리는 방법으로 다차원 행렬 연산에 특화했습니다. 비유하자면 8 코어 CPU가 박사학위를 소지한 수학자 8명이라면, GPU는 (사칙연산밖에 못하지만 계산을 틀리지는 않는) 초등학생 3,000명을 모아놓은 것입니다. 복잡한 수식을 처리한다면 박사들이 유리하겠지만, 초등학교 수학 전집 100권을 빠르게 처리하기에는 초등학생들이 더 유리하겠지요.

딥러닝도 그래픽 연산과 유사하게 다차원 행렬을 계산하기 때문에 GPU를 많이 사용하지만, 최근에는 구글에서 딥러닝 전용으로 만든 TPU(Tensor Processing Unit) 처럼 전용 칩이 개발되고 있습니다.

이 외에 작업에 맞게 직접 설계하여 생산하는 ASIC(Application-Specific Integrated Circuit, 어플리케이션 특화 집적회로) 도 있으며, 그때그때 회로를 직접 프로그래밍해서 사용할 수 있는 FPGA(Field-Programmable Gate Array) 칩도 다양한 산업 분야에서 활용되고 있습니다.

3-15. 기타 (2) 약간의 하드웨어 2: 저장장치 | 10분

3-16. 기타 (3) 클라우드와 가상화 | 10분

3-17. 셸 심화 (1) 편리한 기능 | 10분

3-18. 셸 심화 (2) 셸 스크립트 | 10분

3-19. 셸 심화 (3) 명령어 모음 및 추천자료 | 15분

3-20. 셸 심화 (4) 연습문제 | 40분

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------