

리눅스로 한학기 살기

2020105742 한지훈

1주차: Ubuntu

Concept 1: 비대면 수업 환경 구축하기

2주차: ScreenCloud(수업에서 중요한 부분을 캡처하자)

3주차: Chrome(E-campus 실행해보자)

4주차: Foxit Reader (PDF로 수업 필기하자)

Concept 2: 개발/data science project 실습 환경 구축하기

5주차: Visual Studio Code(python, C++를 editor에서 돌려보자)

6주차: Jupyter Notebook(pandas, matplotlib 등 라이브러리 사용해보자)

7주차: LibreOffice Calc (.csv 형태 dataframe 다뤄보자)

Concept 3: 수업에서 배운 프로그램들 직접 설치해 사용해보기

8주차: Github Desktop(Portfolio 홈페이지를 에디터로 수정해보자)

9주차: Slack(다른 사람들과 협업해보자)

10주차: SnakeViz (프로그램 성능을 한눈에 보자)

11주차: Stacer(고생한 컴퓨터 최적화 해주자)

느낀 점, 소감

<1주차>



➤ 설치 프로그램:

우분투(ubuntu, <https://ubuntu.com/>)

➤ 프로그램 설명:

우분투는 온전한 리눅스 기반 OS로, 기본 원칙은 소프트웨어가 무료이고, 사용자들이 소프트웨어를 필요에 따라 자유롭게 커스텀하여 사용할 수 있다는 것 등이 있다. 데스크톱과 서버용 모두 사용이 가능하며, 워드 프로세서부터 웹서버 소프트웨어, 각종 개발 툴 등 수천가지 소프트웨어를 포함하고 있다.

(출처: <https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html>)

➤ 설치 이유:

소프트웨어 개발방법 및 도구 수업에서 linux 수업을 듣고 나는 어떤 종류의 리눅스 OS를 설치할지, 어떤 방식으로 설치할지 등의 많은 고민을 했다. 우선 OS를 고르는 데에 있어서, 지식이 많지 않았던 나는 상대적으로 범용적이고, 적용 분야가 많다고 하는 우분투를 설치하기로 결정하였다.

➤ 설치 과정:

어떤 OS를 설치할지 정하고 나니 어떤 방식으로 설치할지가 문제로 다가왔다. 가상머신 위에서 돌리자니 성능 문제가 걱정되었고, 윈도우 대신 설치하자니 기존 파일, 호환 프로그램들이 문제였다. 노트북의 용량이 꽤 많이 남아있었기 때문에 이 두 방법의 절충안인 듀얼 부팅 방식을 통해서 우분투를 설치하기로 결정했다.

나의 우분투 설치 과정을 요약하자면 다음과 같다.

1. 우분투 공식 홈페이지에서 iso 이미지 파일을 다운받고,
2. rufus(<https://rufus.ie/>)를 통해서 파일을 usb로 옮겨, 그것을 통해 부팅을 할 수 있도록 준비했다.
3. 노트북 재부팅 후, BIOS로 진입하여(Windows 10의 경우 F2키 연타로 진입함을 알 수 있었다) 부팅 우선순위를 윈도우 boot manager에서 usb로 바꿔준다.
4. 우분투 OS 진입 후에 기타 설정을 하면 설치가 완료됨을 알 수 있었다.

설치 과정은 요약은 했지만, 설치 과정에서 어려웠던 점들이 꽤나 있었다. 우선 iso 파일을 usb로 ‘burn’시키는 과정이 익숙하지 않아서 시간이 좀 걸렸다. 또한 우분투의 default file space가 80G로 지정되었는데, 이것을 100G로 늘리기 위해서 Window file space의 일부를 free space로 분할하고 재설치 하였다. 어찌됐든 우여곡절 끝에 우분투 설치는 성공적으로 마무리 된 듯 하다.

➤ 프로젝트의 목표:

우분투를 설치하고 처음 실행했을 때, 첫 인상은 아주 만족스러웠다. 개인적으로 깔끔한 인터페이스를 좋아하는데, 우분투는 간결함, 그리고 효율성에 초점이 맞춰진 OS라는 느낌이 들었다. 이 때문에 나는 이 프로젝트의 목표를 “완전한 우분투 노트북 만들기”로 잡았다. 현재는 듀얼 부팅 방식으로 윈도우, 우분투가 모두 설치되어 있지만, 내가 윈도우에서 수행하는 작업들을 모두 우분투로 불러와서 우분투 단일체제로 가보고 싶었다. 그래서 앞으로 10주 동안 크게 2가지 종류의 프로그램

1. 일상에 필요한 프로그램(수업 pdf 필기 시 사용하는 annotation 프로그램, chrome 등)
2. 데이터 분석 관련/전반적인 개발 관련 툴(visual studio, R등)

중 10개를 골라 설치할 계획이다. 그리고 이 프로젝트가 끝났을 때 윈도우로 돌아가는 것이 아니라 그 위에 추가 프로그램들을 더 설치하여 더 장기적으로, 혹은 영구적으로 우분투를 main OS로 사용해보고자 한다.

➤ 소감:

우선 설치 후에 신기했던 점 하나는 부팅시에 터미널 같은 창이 뜨고, 거기서 윈도우와 우분투 중 하나를 고를 수 있다는 점이었다. 이전에는 OS위에 돌아가는 프로그램 등만 사용자가 바꿀 수 있다고 생각했는데, OS단계에서 변화를 줄 수 있었다는 것이 인상적이었다.

또한 우분투의 설치 과정에서 Ask Ubuntu(<https://askubuntu.com/>)를 많이 참고하였는데, 글들 중에는 개발자 입장에서 타 OS에서 우분투 등의 리눅스 기반 OS로 바꾸는 것에 우호적인 의견이 많았다. 교수님이 수업 중 해 주신 리눅스 관련 이야기들과 더불어, 장기적으로 우분투를 단일 OS로 가져가려는 나의 결정에 힘을 실어주는 기분이었다. 첫 주차에 우분투를 설치하는 데에 꽤나 시간이 오래 걸렸지만 뿌듯했고, 앞으로 이것 위에서 무엇을 할 수 있을지에 대한 기대가 많이 된다.

<2주차>

➤ 설치 프로그램:

ScreenCloud(<https://screencloud.net/>)

➤ 프로그램 설명:

ScreenCloud는 컴퓨터상의 스크린샷을 생성하고, 공유하는 것을 쉽게 해주는 프로그램이다. 단축키나 아이콘을 이용하여 스크린샷을 찍고, 그것을 개인 서버나 클라우드로 업로드할 수 있게 해준다. 또한 클립보드에 링크가 자동 생성되어 사진을 공유하기에 용이하다.

(출처: <https://screencloud.net/>)

➤ 설치 이유:

우선 이 프로젝트에서 주차별로 설치 과정을 담은 사진을 우분투에서 담기 위해서는 별도의 캡처 프로그램이 필요하다고 생각했다. 또한 기타 쓰임새가 많고, 설치가 비교적 간단할 것 같아서 첫 프로그램으로 정하였다.

➤ 설치 과정:

우선 해당 프로그램을 설치하는 방법을 구글링 하였다. 찾아보니 (<https://snapcraft.io/install/screencloud/ubuntu>) Snap을 통해 설치가 가능하다고 설명이 되어있다.

Enable snaps on Ubuntu and install ScreenCloud

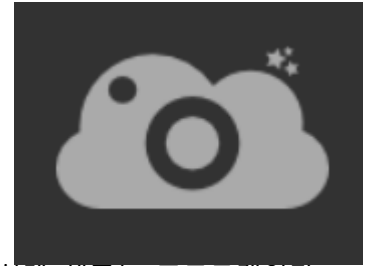
Snaps are applications packaged with all their dependencies to run on all popular Linux distributions from a single build. They update automatically and roll back gracefully.

Snaps are discoverable and installable from the [Snap Store](#), an app store with an audience of millions.

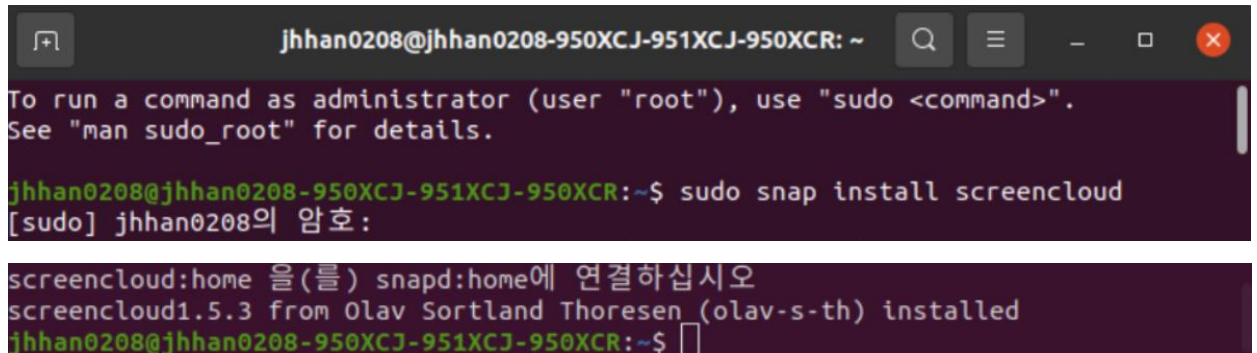
Snap에 대해 알아보니 리눅스 기반 OS에서 돌아갈 수 있는 여러 어플리케이션들을 묶어 놓은 것이라고 볼 수 있었다. 구글 플레이스토어나 크롬 웹스토어 등 어플 설치의 편의성을 위해 만들어진 것과 비슷한 것 같았다.

터미널과 친해지는 것이 중요하다고 생각해서, 이 페이지에 나와있는 대로 터미널을 통해서 설치하기로 하였다. 다음 명령어를 통해서 설치가 가능하다고 말한다.

```
$ sudo snap install screencloud
```



명령어를 봤을 때 우선 뒤의 세 단어를 통해 “snap을 통하여 screencloud를 설치해라”정도의 의미를 생각할 수 있었는데, 앞의 sudo라는 단어는 무슨 뜻인지 알지 못했다. 우선 터미널을 켜보기로 했다. “Ctrl + Alt + T”를 통하여 터미널을 실행하니 맨 위에 관리자 권한으로 명령을 실행할 때 “sudo <command>” 형태로 실행하라고 나와있었다. 총합해보면, 이 줄의 뜻은 “관리자 권한으로 snap을 통해 screencloud를 설치해라” 이라는 것을 알게 되었다. 줄을 입력하고 실행하니 금방 screencloud가 설치되는 것을 볼 수 있었다.

A terminal window with a dark purple background. The title bar shows the user 'jhhan0208' and the host 'jhhan0208-950XCJ-951XCJ-950XCR'. The terminal text includes a hint about using 'sudo' for administrative commands, followed by the command 'sudo snap install screencloud'. It prompts for the user's password, which is masked with asterisks. After successful installation, it shows the version 'screencloud 1.5.3' and the maintainer 'Olav Sortland Thoresen (olav-s-th)'.

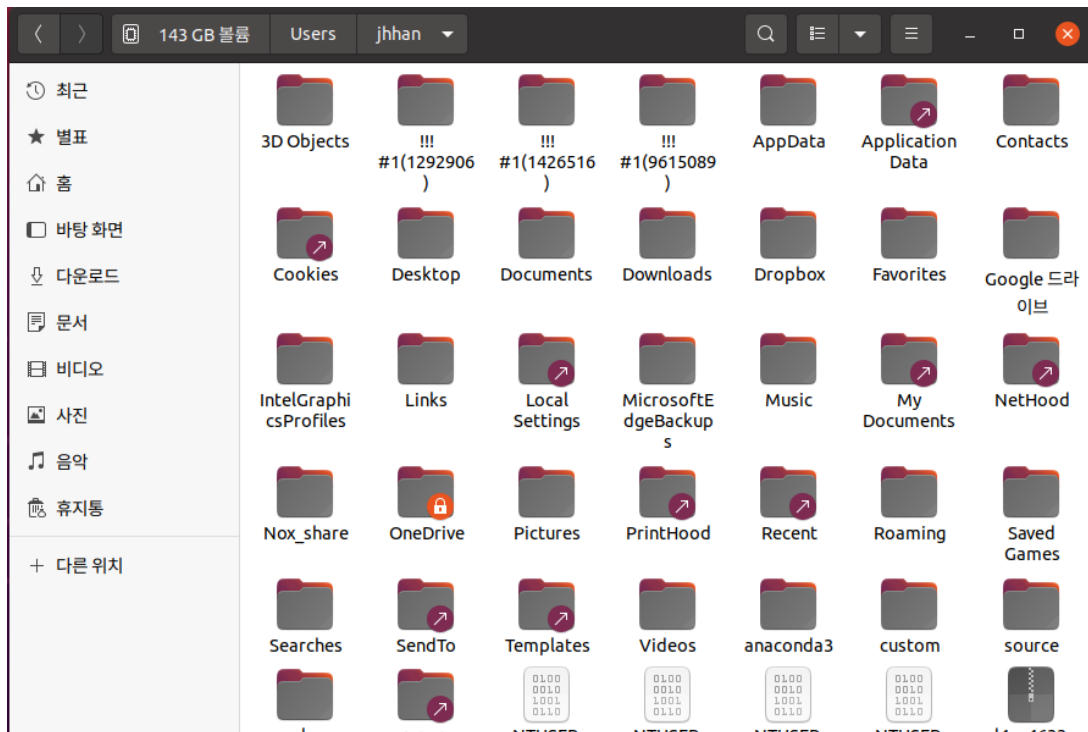
```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR: ~  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo snap install screencloud  
[sudo] jhhan0208의 암호 :  
  
screencloud:home 을 (를) snapd:home에 연결하십시오  
screencloud1.5.3 from Olav Sortland Thoresen (olav-s-th) installed  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$
```

➤ 사용/느낀 점:

우선 screencloud를 통해 설치 중 터미널 등 설치 과정을 담은 화면들을 캡처해봤다. 그리고 다른 기능인 전체화면 캡처, 활성화된 창 캡처 등도 해보았다. 각 기능에 대해 단축키가 있어서(각각 Shift + Alt + 1/2/3) 각 기능을 사용하기가 매우 편리했다. 또한 찍은 후에 바로 저장할 폴더를 물어보고 빠르게 저장할 수 있다. 전반적으로 매우 만족했고, 어떤 면에서는 오랫동안 사용했던 윈도우의 캡처도구보다 편리하다고 느꼈다.

사용 중 화면을 직접 찍지는 못했다. 캡처 도구를 사용하는 중에 그 화면 자체를 캡처할 수는 없기 때문이다. 이 때문에 사용 중 사진은 다음주부터 첨부해야 할 것 같다.

캡처한 사진을 윈도우로 옮기려는 과정에서 흥미로운 것을 발견했다. 바로 우분투에서 윈도우 파일에 직접 접근할 수 있다는 것이다.



파일 > 다른 위치 > users > jhhan(사용자 이름) 으로 가면 Desktop 폴더를 통해 바탕화면의 파일에도 액세스할 수 있고, 다운로드 등 모든 윈도우 파일을 직접 열람하는 것이 가능하다. 윈도우와 우분투의 file space는 완전히 단절된 줄 알았는데, 파일에 손쉽게 접근하고 수정할 수 있다는 것이 신기했다.

요약하자면, 이번주에는 screencloud를 설치해보았다. 그 과정에서 snap이 무엇인지, 간단한 터미널 명령어, 그리고 우분투에서 윈도우 파일에 직접 접근할 수 있다는 것을 배웠다. 이 프로젝트의 첫번째 프로그램을 성공적으로 설치해서 뿌듯하고, 앞으로도 더 많은 프로그램들을 통해 이 노트북을 바꿔나가고 싶다.

<3주차>



➤ 설치 프로그램:

Google Chrome(<https://www.google.com/intl/ko/chrome/>)

➤ 프로그램 설명:

크롬은 구글에서 제공하는 오픈소스 웹 브라우저이다. 기존 브라우저보다 더 뛰어난 속도, 안정성, 보안성을 갖추는 것이 목표이다. 실제로 메모리 사용량을 최적화하여 높은 속도를 보여주며, 위험한 사이트 차단, 주기적인 업데이트 등으로 안전성과 보안성을 가지고 있다. 또한 윈도우/리눅스/안드로이드 등 여러 운영체제에 모두 돌아갈 수 있다는 것이 장점이다.

(출처: <https://www.google.com/intl/ko/chrome/productivity/>)

➤ 설치 이유:

컴퓨터를 사용함에 있어서 웹 브라우저의 사용은 필수적이기 때문에 우분투 설치 후에 가장 먼저 살펴본 부분이 웹 브라우저였고, 우분투의 경우 default로 Firefox라는 웹 브라우저가 설치 되어있다는 것을 알 수 있었다. 이왕이면 우분투에서 기본으로 제공하는 Firefox를 써보고자 하였으나, 전에 사용하던 Chrome에 비해서 불편한 점이 많았다. 특히 여러 사이트의 아이디/비밀번호가 Chrome상에 저장되어 있고, 웹스토어의 프로그램을 몇 개 사용하는 등 Chrome의 많은 기능들을 활용하고 있어서 이것을 바꾸기가 쉽지 않았다. 그래서 결국에는 Chrome을 우분투 위에도 설치해보기로 결정했다.

➤ 설치 과정:

Chrome의 설치 방법을 찾아보던 중, Linuxize(<https://linuxize.com/>)라는 유용한 사이트를 알게 되었다. 둘러보니 우분투 뿐만 아니라 각종 운영체제에 프로그램들을 설치하는 법, 응용법 등이 많이 나와있었다. 아마 앞으로 프로젝트를 진행하면서 이 사이트를 더 참고할 것 같다. 어쨌든 오늘은 이 사이트대로 Chrome을 설치해보기로 하였다.

터미널 사용에 조금이라도 더 익숙해지기 위해 줄 별로 의미를 알아보았다.

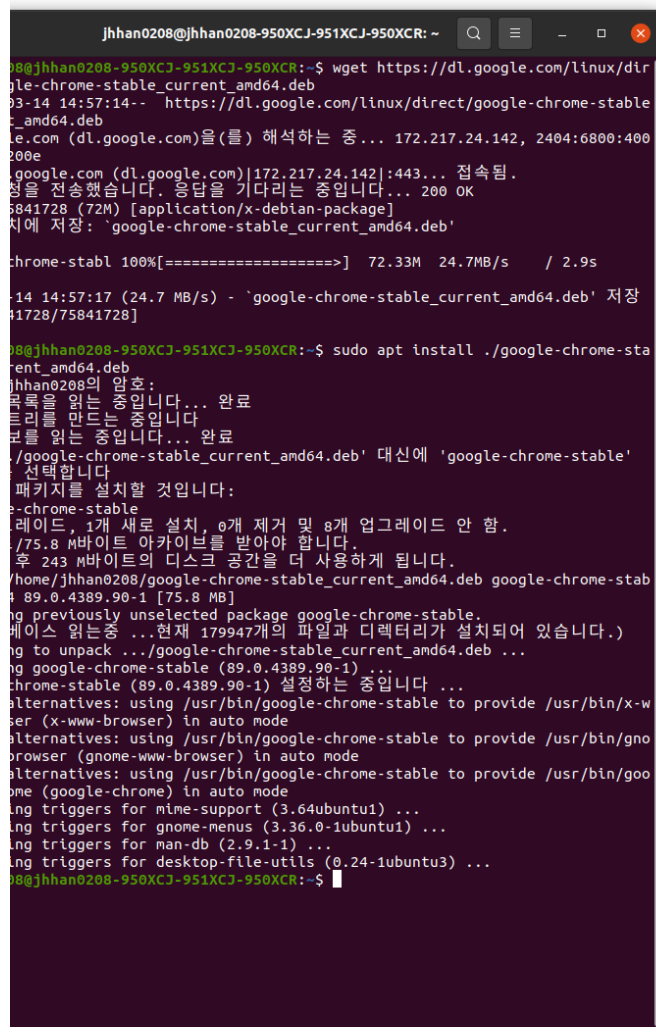
```
$ wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

wget를 통해 크롬 .deb 파일을 설치한다고 설명이 나와있다. Wget은 리눅스 명령어의 하나인 Web Get의 약자로, 웹 상의 파일을 다운로드 받을 때 사용한다고 한다. 바로 뒤에 https로

시작하는 웹 주소로부터 프로그램을 가져온다고 나와있어서 이해가 되었다. .deb는 소프트웨어 패키지 포맷의 확장자임을 알게 되었다. 즉 “해당 주소로 찾아가서 .deb 형태의 설치파일을 가져와라”는 말의 줄임이라는 것을 배웠다.

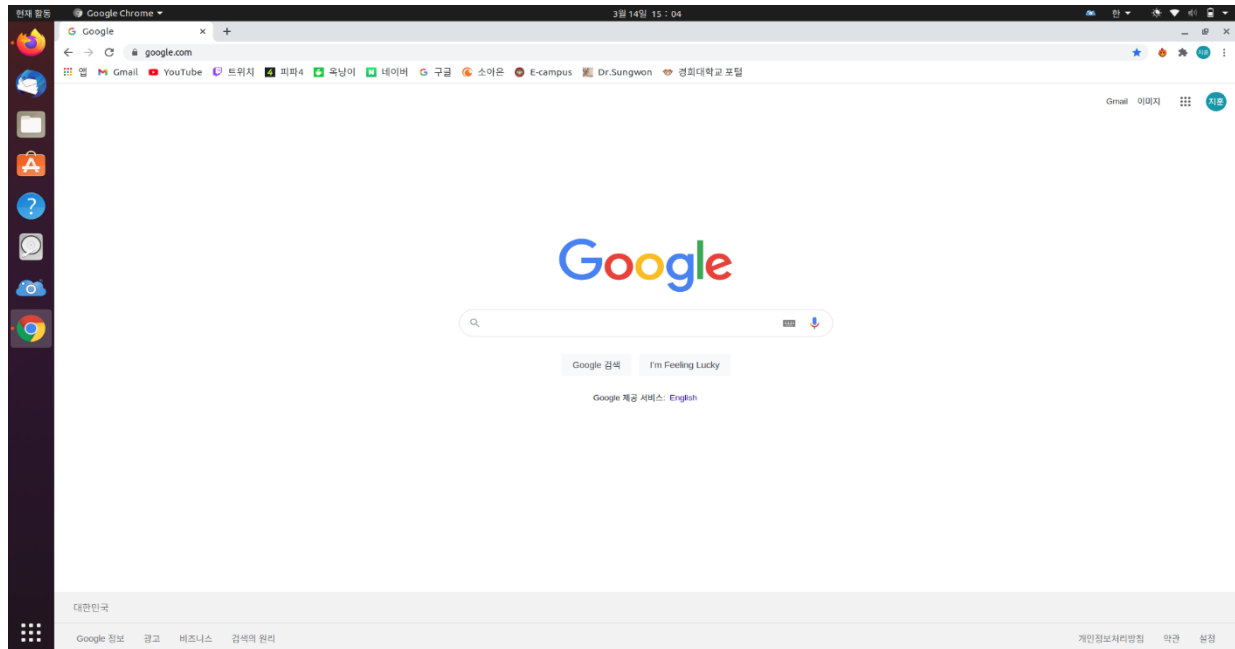
```
$ sudo apt install ./google-chrome-stable_current_amd64.deb
```

이 줄에서는 apt가 무슨 의미인지 몰라 찾아보았다. Apt는 패키지 관리 툴을 사용할 때 사용하는 명령어로, ‘apt install’, ‘apt remove’ 등의 형태로 묶어서 사용한다고 한다. 저번주에 배운 **sudo**와 결합하면, “관리자 권한으로 가져온 패키지 파일을 설치해라” 정도의 의미가 되는 것 같다. 터미널에 입력하니 정상적으로 설치가 되었음을 볼 수 있었다.



```
jghan0208@jghan0208-950XCJ-951XCJ-950XCR: ~  
jghan0208@jghan0208-950XCJ-951XCJ-950XCR:~$ wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb  
2023-14 14:57:14-- https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb  
Resolving dl.google.com (dl.google.com)을(를) 해석하는 중... 172.217.24.142, 2404:6800:4002:00e00000:800e  
Connecting to dl.google.com (dl.google.com)|172.217.24.142|:443... 접속됨.  
HTTP/2 200 OK  
Content-Length: 72M [application/x-debian-package]  
Content-Type: application/x-debian-package  
에 저장: 'google-chrome-stable_current_amd64.deb'  
google-chrome-stable 100%[=====] 72.33M 24.7MB/s / 2.9s  
2023-14 14:57:17 (24.7 MB/s) - 'google-chrome-stable_current_amd64.deb' 저장됨 [1728/75841728]  
jghan0208@jghan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install ./google-chrome-stable_current_amd64.deb  
jghan0208의 암호:  
패키지를 읽는 중입니다... 완료  
트리거를 만드는 중입니다  
패키지를 읽는 중입니다... 완료  
./google-chrome-stable_current_amd64.deb 대신에 'google-chrome-stable'을 선택합니다  
패키지를 설치할 것입니다:  
google-chrome-stable  
레이드, 1개 새로 설치, 0개 제거 및 8개 업그레이드 안 함.  
75.8 MB의 아카이브를 받아야 합니다.  
또 243 MB의 디스크 공간을 더 사용하게 됩니다.  
./home/jghan0208/google-chrome-stable_current_amd64.deb google-chrome-stable 89.0.4389.90-1 [75.8 MB]  
Previously unselected package google-chrome-stable.  
패키지 읽는 중 ... 현재 17994개의 파일과 디렉터리가 설치되어 있습니다.)  
unpacking ./google-chrome-stable_current_amd64.deb ...  
unpacking google-chrome-stable (89.0.4389.90-1) ...  
google-chrome-stable (89.0.4389.90-1) 설정하는 중입니다 ...  
alternatives: using /usr/bin/google-chrome-stable to provide /usr/bin/x-www-browser (x-www-browser) in auto mode  
alternatives: using /usr/bin/google-chrome-stable to provide /usr/bin/gnome-browser (gnome-www-browser) in auto mode  
alternatives: using /usr/bin/google-chrome-stable to provide /usr/bin/google-chrome (google-chrome) in auto mode  
Setting up triggers for mime-support (3.64ubuntu1) ...  
Setting up triggers for gnome-menus (3.36.0-1ubuntu1) ...  
Setting up triggers for man-db (2.9.1-1) ...  
Setting up triggers for desktop-file-utils (0.24-1ubuntu3) ...  
jghan0208@jghan0208-950XCJ-951XCJ-950XCR:~$
```

설치 후에는 구글 계정 로그인 등의 간단한 절차 이후에, 우분투에서 크롬을 이용할 수 있게 되었다! 잘 동작하는지 확인하기 위해 구글을 실행해 보았다.



```
sudo apt-get purge firefox
```

이 줄로 어플리케이션을 지운다고 나와있다. 앞서 apt에 대해 살펴봤는데, apt-get에 대해서도 알아보니 각각 패키지 관리 툴인 'apt-get'과 'apt-cache'의 명령을 결합하여 'apt' 형태로 쓴다고 한다. 즉 핵심 기능 외에 기타 세부기능을 사용할 때는 apt-get를 사용한다고 한다. apt-get purge는 apt-get 명령어의 하나로, 패키지와 관련 툴을 제거하는 역할을 한다. 이 경우에는 firefox 어플리케이션을 제거하기 위해 쓰였다.

```
sudo rm -Rf /etc/firefox/
```

```
sudo rm -Rf /usr/lib/firefox*
```

위의 두 줄은 firefox가 설치되어 있었던 기타 폴더들을 제거하는 역할을 한다고 한다. 마찬가지로 정보를 찾아보니, rm은 remove의 약어임을 알 수 있었다. 여기에 -r을 붙이면 폴더에 접근할 수 있게 되고, -f를 붙이면 강제로 명령을 실행한다. 종합하자면 관리자 권한으로 설치 폴더를 강제로 지워주는 명령어인 것이다. 이 줄들을 실행하니 무사히 firefox를 지울 수 있었다.

```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR: ~  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt-get purge firefox  
[sudo] jhhan0208의 암호:  
패키지 목록을 읽는 중입니다... 완료  
의존성 트리를 만드는 중입니다  
상태 정보를 읽는 중입니다... 완료  
다음 패키지를 지울 것입니다:  
firefox*  
0개 업그레이드, 0개 새로 설치, 1개 제거 및 8개 업그레이드 안 함.  
이 작업 후 222 M바이트의 디스크 공간이 비워집니다.  
계속 하시겠습니까? [Y/n] Y  
(데이터베이스 읽는중 ...현재 180060개의 파일과 디렉터리가 설치되어 있습니다.)  
firefox (86.0+build3-0ubuntu0.20.04.1)를 제거합니다...  
Processing triggers for mime-support (3.64ubuntu1) ...  
Processing triggers for hicolor-icon-theme (0.17-2) ...  
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...  
Processing triggers for man-db (2.9.1-1) ...  
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...  
(데이터베이스 읽는중 ...현재 179971개의 파일과 디렉터리가 설치되어 있습니다.)  
Purging configuration files for firefox (86.0+build3-0ubuntu0.20.04.1) ...  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo rm -Rf /etc/firefox/  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo rm -Rf /usr/lib/firefox*  
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$
```

여기서 `rm -Rf` 명령어 자체를 추천하지는 않는다는 글들을 접할 수 있었다. 우선 강제로 삭제하는 과정에서 어떤 오류가 발생하는지 알 수 없고, 이 명령어에 `*`을 실수로 붙여버리면 해당 디렉토리의 모든 파일까지 날라갈 수 있다는 등의 이유 때문이었다. 앞으로는 명령어를 그것이 초래할 수 있는 결과를 고려해가면서 사용해야 한다는 것을 느꼈다.

➤ 사용/느낀 점:

우선 나는 크롬을 통해 웹서핑 외에도 메일, 방송 시청 등을 많이 하는 편이고, 크롬의 응용프로그램들도 많이 사용하고 있기 때문에 크롬 설치 하나로도 뭔가 인터페이스 면에서 우분투가 훨씬 익숙해지는 느낌을 받았다.

터미널을 사용하는 것도 재미있다. 기본적으로는 명령어를 구글링 하지만, 그것을 그대로 복사하여 실행하기보다는 각 키워드의 뜻을 알아가는 것이 유익했다. 마치 파이썬, C++ 등을 처음 배울 때 함수, 객체 등의 개념들을 하나씩 배워가는 것이라 비슷한 느낌이 들었다.

3주정도를 진행하니 보고서 구성에 대한 감이 어느정도 잡힌 것 같다. 앞으로 몇 주동안은 계속해서 ‘우분투 사용자’ 입장에서 이용을 더 편리하게 해주는 프로그램들을 설치할 계획이다. 그리고 이후에는 ‘개발자’ 입장에서 우분투를 활용하여 할 수 있는 데이터 분석/개발 관련 툴들을 설치해나갈 것이다.

<4주차>



➤ 설치 프로그램:

Foxit Reader(<https://www.foxitsoftware.com/pdf-reader/>)

➤ 프로그램 설명:

Foxit Reader는 PDF 파일을 읽을 수 있게 해주는 응용 어플리케이션이다. PDF 열람, 인쇄 등을 지원하고, 주석을 달거나 하는 등 문서를 수정할 수도 있다. 이에 더불어 여러 사람과 공동작업을 할 수 있고, 링크를 통해 공유도 원활하게 해준다.

(출처: <https://www.foxitsoftware.com/pdf-reader/>)

➤ 설치 이유:

리눅스를 더 자주 사용해보기 위해서 학교 수업도 우분투에서 들어보기로 했다. 나는 보통 교수님들이 제공하시는 pdf 파일을 다운로드하여 PDF 편집 프로그램으로 밑줄을 치거나 주석을 다는 방법으로 수업을 듣는 편이다. 윈도우에서는 ezPDF Editor 3.0(<http://www.ezpdf.co.kr/editor3/main.do>) 프로그램을 이용 했었는데, 이 프로그램은 리눅스는 지원하지 않기 때문에 새로운 편집 프로그램을 찾아보기로 했다. 여러 프로그램들을 찾아본 결과 UI도 깔끔하고 필기 관련 다양한 기능을 지원하는 Foxit Reader를 설치해보기로 했다.

➤ 설치 과정:

이번주도 마찬가지로 터미널을 통해 설치해보기로 했다. 이번주는 LinuxBabe(<https://www.linuxbabe.com/desktop-linux/install-foxit-pdf-reader-ubuntu>)의 도움을 받기로 했다. 터미널 명령어를 한 줄씩 살펴보았다.

```
wget http://cdn01.foxitsoftware.com/pub/foxit/reader/desktop/linux/2.x/2.4/en_us/FoxitReader.enu.setup.2.4.4.0911.x64.run.tar.gz
```

줄 맨앞에 반가운 wet가 보였다. wet는 Web Get의 약자로, 웹 상의 파일을 다운로드 받을 때 사용한다는 것을 저번주에 배웠다. 따라서 이 줄은 wet을 이용하여 Foxit Reader의 64비트 버전 설치파일을 다운받는 줄임을 알 수 있었다.

```
cd ~/Downloads
```

이 줄의 `cd`는 `change directory`의 약자로, 파일 디렉토리를 변경할 때 쓰는 명령어임을 찾았다. `cd` 명령어로 `Downloads` 폴더로 이동하라는 의미인 것이다. 그런데 나의 경우는 설치파일이 `Home directory` 바로 아래에 설치되었기 때문에 이 줄을 실행시킬 필요가 없었다.(이 사실을 알아채기까지 꽤나 시간이 걸렸다.)

```
tar xzvf FoxitReader*.tar.gz
```

`tar`이라는 명령어는 처음 접했다. 찾아보니 `tar` 명령어는 리눅스에서 파일을 압축하거나 압축을 푸는데에 쓰인다고 한다. 이 뒤에 오는 명령어(여기서는 `xzvf`)로 여러 기능을 수행할 수 있다고 한다. 확장자명이 `.tar`인 파일의 압축을 풀 때는 `xvf`를 붙이고, 확장자명이 `.tar.gz`인 파일의 압축을 풀 때는 이것에 `z`를 붙여서 `xzvf`를 붙이면 된다. 따라서 다운로드 받은 파일의 압축을 풀어 실행파일로 만드는 명령어인 것이다.

```
sudo chmod a+x FoxitReader*.run
```

이 줄에서 처음 본 `chmod`는 각종 권한 관리에 대한 명령어라는 것을 알게 되었다. 뒤에 왜 갑자기 `a+x` 라는 수식이 나오는지 의문이었지만, 구글링을 해보니 의미를 알 수 있었다. 여기서 `a`는 모든 사용자, `+`는 권한 추가, `x`는 실행권한이라는 뜻이었다. 조합해보면 ‘모든 사용자에게 실행권한을 추가해라’의 의미가 됨을 알 수 있었고, 이 외에도 `-`, `=` 그리고 다양한 기호들이 있다는 것을 배웠다. 즉 실행파일을 실행할 수 있는 권한을 나에게 부여하는 줄이었다.

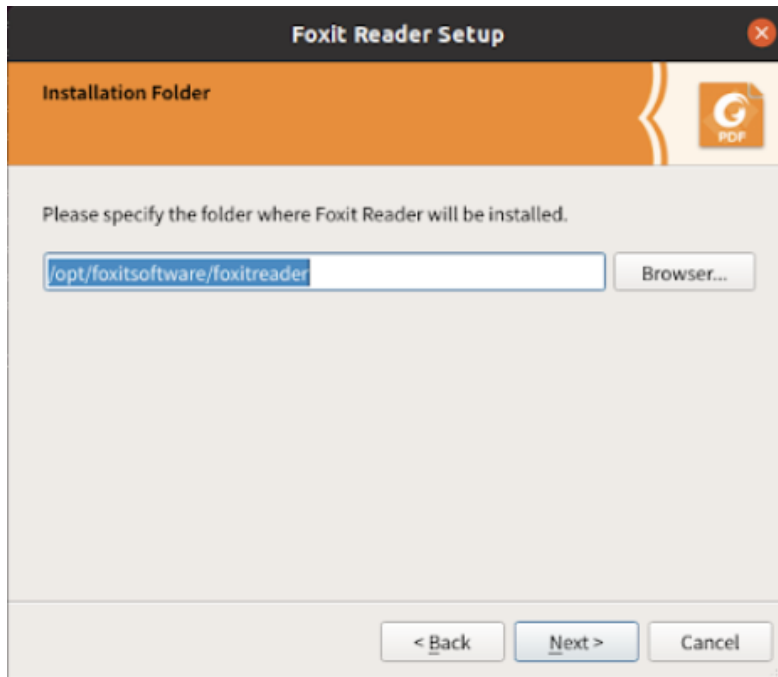
```
sudo ./FoxitReader*.run
```

이 파일은 단순히 관리자 권한으로 받은 파일을 실행하는 줄이라는 것을 바로 알 수 있었다. 지금까지의 줄을 모두 터미널에서 실행해보았다.

```
208-950XCJ-951XCJ-950XCR:~$ wget http://cdn01.foxitsoftware.com/pub/foxit/reader/desktop/linux/2.x/2.4/en_us/FoxitReader.enu.setup.2.4.4.0911.x64.run.tar.gz
32:24-- http://cdn01.foxitsoftware.com/pub/foxit/reader/desktop/linux/2.x/2.4/en_us/FoxitReader.enu.setup.2.4.4.0911.x64.run.tar.gz
are.com (cdn01.foxitsoftware.com)을(를) 해석하는 중... 64.62.208.5, 64.62.208.6, 184.105.233.248, ...
software.com (cdn01.foxitsoftware.com)[64.62.208.5]:80... 접속됨.
했습니다. 응답을 기다리는 중입니다... 200 OK
70M) [application/x-gzip]
: 'FoxitReader.enu.setup.2.4.4.0911.x64.run.tar.gz'
```

```
208-950XCJ-951XCJ-950XCR:~$ cd ~/Downloads
jhhan0208/Downloads: 그런 파일이나 디렉터리가 없습니다
208-950XCJ-951XCJ-950XCR:~$ cd ~/Download
jhhan0208/Download: 그런 파일이나 디렉터리가 없습니다
208-950XCJ-951XCJ-950XCR:~$ tar xzvf FoxitReader*.tar.gz
setup.2.4.4.0911(r057d814).x64.run
208-950XCJ-951XCJ-950XCR:~$ sudo chmod a+x FoxitReader*.run
의 암호:
208-950XCJ-951XCJ-950XCR:~$ sudo ./FoxitReader*.run
```

마지막 줄을 실행하니 아래처럼 GUI가 나오고 각종 설정 이후 설치가 성공적으로 되었음을 알 수 있었다.

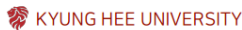


➤ 사용/느낀 점:

우선 해당 소프트웨어 개발방법 및 도구 수업 ppt를 Foxit Editor로 열어보고, 필기도 해봤다. 전반적으로 만족스러운 프로그램이었다. 내가 자주 사용하는 단어 강조, 영역 강조, 밑줄, text 입력 기능과 추가적인 많은 기능이 있었고, 사용하기에도 간단했다. 아래 사진은 필기해본 하나의 페이지이다.

Class

- **Data analytics** (Data Analysis, Data Science)
 - is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making
 - Preferred languages : Why?
 - **Python**
 - Python:
수학자, 과학자 등 여러분야의 사람들 연구에 활용
-> 수학, 물리, 통계 관련 라이브러리가 많음
 - **R**
 - > 시간이 지나면서 데이터쪽에 자연스럽게 강세 보임
 - Major requirements
 - Natural language processing, Syntax processing, etc.
문자열 처리(python이 강함)
 - **Linear algebra, Probability, Statistics**



앞으로 수업은 우분투에서 듣기로 결정했다. 그리고 이번 기회에 좀 더 많은 사람들이 이용하는 PDF 편집 프로그램으로 바꾼 것에 의의가 있는 것 같다.

터미널에 대한 지식도 계속 늘어나는 기분이라서 좋다. 특히 chmod의 다양한 연산자로 여러 권한을 부여하는 부분이 흥미로웠다. 주마다 프로그램을 우분투에 설치하는 것이 마치 새 집에 가구들을 하나씩 들이는 기분이다.

<5주차>



➤ 설치 프로그램:

Visual studio code(<https://code.visualstudio.com/>)

➤ 프로그램 설명:

비주얼 스튜디오 코드는 가볍지만 강력한 소스코드 에디터이다. 윈도우, mac, 리눅스 등 다양한 운영체제와 호환이 되며, C++ python, Java 등 다양한 언어를 지원한다. 오픈 소스이며 범용성이 높다는 것이 특징이다.

➤ 설치 이유:

지난주까지 해서 수업을 듣기 위한 리눅스 세팅은 완성되었다고 생각한다. 따라서 이번주부터는 좀 더 프로그래밍 언어, 데이터 분석 등 전공과 관련 있는 테마의 프로그램들을 설치하기로 했다. 우선 저번 학기에 객체 지향 프로그래밍 수업을 들은 이후로 C++로 알고리즘 문제들을 풀어보고 있는데, 이때 소스코드를 실행할 프로그램이 필요했다. 또한 이 수업에서 python을 나의 language로 선택했고, 이후 DS 프로젝트 등에 python을 쓰기 때문에 소스코드 에디터가 매우 필요했다. 여러 에디터 중 리눅스 등 다양한 OS를 지원하고 mainstream이라 할 수 있는 Visual studio code를 선택했다.

➤ 설치 과정:

이번에도 Linuxize(<https://linuxize.com/>)을 참고하였다. 3주차 크롬을 설치할 때에도 도움이 되었는데, 정말 리눅스에 관한 자료가 풍부하다는 것을 느꼈다. 이번주도 어김없이 명령어들을 한 줄씩 뜯어보기로 했다.

```
$ sudo apt update
$ sudo apt install software-properties-common apt-transport-https wget
```

우선 이 두 줄은 설치에 앞서서 Package index를 업데이트하고 필요한 종속적인 파일들을 다운로드 받는 역할이라고 한다. Apt는 패키지 관리 툴에 관한 명령어였으므로 첫째줄은 패키지 툴을 업데이트하라는 내용일 것이다. 두번째 줄을 보니 뜻을 모르겠는 “software-properties-common”과 “apt-transport-https”를 설치하라고 한다. 각각 무슨 뜻일지를 알아보았다.

우선 “software-properties-common”는 사용자가 소프트웨어를 다운로드 받는 repository들에

대한 정보를 제공하는 것이라고 한다. 그래서 어떤 소스 들로부터 파일들을 받는지에 대한 것을 알 수 있게 해준다. “apt-transport-https”는 보안이 강한 https 프로토콜로 다운로드를 하게 해주는 명령어이다. 두 줄 모두 설치의 기반으로써 필요하다는 것을 느꼈다.

```
$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add
```

이 줄은 Microsoft GPG key를 불러오는 줄이라고 한다. Microsoft GPG key는 무엇일까? GPG key에 대해 알아보니 public key와 private key를 이용하여 정보, 메시지, 프로그램 등을 보내고 받을 때 보안을 유지하기 위한 암호학 패키지라고 한다. 이 경우에는 Microsoft의 public key를 불러옴으로써 프로그램의 소스가 안전한지 판단하는 것이다.

```
$ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos,
```

이 줄은 visual studio code를 설치할 repository를 추가하는 명령어다. “Apt를 통하여 해당 이름의 repository를 add해라”라는 의미로 직관적으로 이해할 수 있었다.

```
$ sudo apt install code
```

드디어 마지막 줄은 visual studio code 패키지를 설치하는 명령어다. 마찬가지로 그 의미를 줄 안에서 쉽게 찾아볼 수 있었다. 모든 줄을 실행하니 성공적으로 visual studio code가 설치된 것을 볼 수 있었다.


```

jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR: ~
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt update
[sudo] jhhan0208의 암호:
기존:1 https://download.docker.com/linux/ubuntu focal InRelease
기존:2 http://dl.google.com/linux/chrome/deb stable InRelease
기존:3 http://kr.archive.ubuntu.com/ubuntu focal InRelease
기존:4 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease
기존:5 http://security.ubuntu.com/ubuntu focal-security InRelease
기존:6 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
패키지 10이(가) 업그레이드되었습니다. 'apt list --upgradable'를 실행하여 확인해 보십시오.
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install software-properties-common apt-transport-https wget
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
패키지 wget는 이미 최신 버전입니다 (1.20.3-1ubuntu1).
wget 패키지는 수동설치로 지정합니다.
패키지 software-properties-common는 이미 최신 버전입니다 (0.98.9.4).
software-properties-common 패키지는 수동설치로 지정합니다.
패키지 apt-transport-https는 이미 최신 버전입니다 (2.0.4).
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  linux-headers-5.8.0-44-generic linux-hwe-5.8-headers-5.8.0-44
  linux-image-5.8.0-44-generic linux-modules-5.8.0-44-generic
  linux-modules-extra-5.8.0-44-generic
'sudo apt autoremove'를 이용하여 제거하십시오.
0개 업그레이드, 0개 새로 설치, 0개 제거 및 10개 업그레이드 안 함.
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
OK
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
기존:1 https://download.docker.com/linux/ubuntu focal InRelease
기존:2 http://dl.google.com/linux/chrome/deb stable InRelease
받기:3 https://packages.microsoft.com/repos/vscode stable InRelease [3,959 B]
기존:4 http://kr.archive.ubuntu.com/ubuntu focal InRelease
받기:5 https://packages.microsoft.com/repos/vscode stable/main amd64 Packages [233 kB]
기존:6 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease
기존:7 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease
기존:8 http://security.ubuntu.com/ubuntu focal-security InRelease
내려받기 237 k바이트, 소요시간 1초 (214 k바이트/초)
패키지 목록을 읽는 중입니다... 완료

```

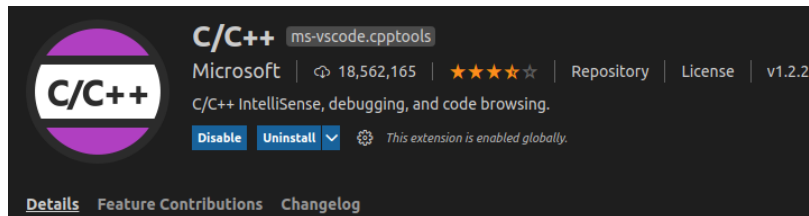
```

jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install code
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:
  linux-headers-5.8.0-44-generic linux-hwe-5.8-headers-5.8.0-44
  linux-image-5.8.0-44-generic linux-modules-5.8.0-44-generic
  linux-modules-extra-5.8.0-44-generic
'sudo apt autoremove'를 이용하여 제거하십시오.
다음 새 패키지를 설치할 것입니다:
  code
0개 업그레이드, 1개 새로 설치, 0개 제거 및 10개 업그레이드 안 함.
71.9 M바이트 아카이브를 받아야 합니다.
이 작업 후 282 M바이트의 디스크 공간을 더 사용하게 됩니다.
받기:1 https://packages.microsoft.com/repos/vscode stable/main amd64 code amd64 1.54.3-1615806378 [71.9 MB]
내려받기 71.9 M바이트, 소요시간 11초 (6,576 k바이트/초)
Selecting previously unselected package code.
(데이터베이스 읽는중 ...현재 217727개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../code_1.54.3-1615806378_amd64.deb ...
Unpacking code (1.54.3-1615806378) ...
code (1.54.3-1615806378) 설정하는 중입니다 ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for shared-mime-info (1.15-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$

```

➤ 사용/느낀 점:

설치 이후에는 C/C++ 툴을 설치하고 컴파일러 등 설정을 좀 하니 소스코드를 실행할 수 있는 환경을 리눅스에서 구축할 수 있었다.



객체지향 프로그래밍 수업 중 Class에 대해 배웠던 파일을 가져와서 실행해보았다.

```
class.c++ - c++ (Workspace) - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
C++ (WORKSPACE)
  > .vscode
  > backtracking
  > backtracking.c++
  > c++.code-workspace
  > class
  > class.c++
  > jungol
C++ > class.c++ > ...
1  using namespace std;
2  #include <iostream>
3  #include <string>
4  #include <iomanip>
5
6
7  // class 만드는 법
8  // 1) class name{};
9  // 2) private:, public: 영역 분할
10 // 3) member data 만들기 in private
11 // 4) method 만들기 in public
12 // 5) 초기화 함수(constructor, 생성자)
13 // - 생성자 이름은 class 이름과 동일
14 // - 만약 내가 생성자를 만들지 않으면, default 생성자(Point()){}가 자동으로 생성
15 // - 만약 내가 생성자를 만들면, default 생성자가 자동 생성 안됨 > 직접 만들어줘야 함
16
17 class Point { // data 와 function으로 부분(각각 private, public)
18 private: // 외부(main)에서 접근 불가능, member data
19     int x;
20     int y;
21 public: // 외부(main)에서 접근 가능, member function(method)
22
23     Point(int _x, int _y) { // 생성자(클래스 이름과 동일, 객체 생성시 한번만 실행)
24         x = _x;
25         y = _y;
26     }
27
28     Point(){} // default 생성자
29
30     int getX() { return x; }
31     int getY() { return y; }
32     void setXY(int _x, int _y) {
33         x = _x;
34         y = _y;
35     }
36     void print() {
```

```
22
23 Point(int _x, int _y) { // 생성자(클래스 이름과 동일, 객체 생성시 한번만 실행)
24     x = _x;
25     y = _y;
26 }
27
28 Point(){} // default 생성자
29
30 int getX() { return x; }
31 int getY() { return y; }
32 void setXY(int _x, int _y) {
33     x = _x;
34     y = _y;
35 }
36 void print() {
37     cout << "(" << x << ", " << y << ")" << endl;
38 }
39 };
40
41 int main() {
42
43     Point pt1; // 클래스의 인스턴스인 객체 선언
44     //pt1.x = 1;
45     //pt1.y = 1; 이 두줄은 실행 안됨(변수가 private)
46     pt1.setXY(1, 2);
47
48     cout << pt1.getX() << endl;
49     cout << pt1.getY() << endl;
50
51     pt1.print();
52
53     Point pt2(10, 20);
54     pt2.print();
55
56     return 0;
57 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[Running] cd "/home/jhhan0208/바탕화면/c++/" && g++ class.cpp -o class && "/home/jhh
1
2
(1, 2)
(10, 20)

[Done] exited with code=0 in 0.386 seconds
```

결과가 정상적으로 실행되는 것을 볼 수 있었다. 수업에서 배운 내용을 바탕으로 백준 알고리즘 사이트(<https://www.acmicpc.net/>)에서 알고리즘 문제들을 풀었었는데, 이제 우분투에서 하면 되겠다고 생각했다. 또한 Visual Studio Code는 여러 언어를 지원하기 때문에 나중에 Python 툴도 설치하여 웹파이선 프로그래밍 수업 내용도 복습하고, 새로운 코드를 짜는 데에도 활용해보야겠다는 생각을 하였다.

이번주에 처음으로 기본 응용 프로그램이 아닌 개발자로서 필요한 도구를 설치해봤는데, 성공적으로 C++ 소스코드 실행환경을 우분투에서 성공적으로 구축할 수 있어서 뿌듯했다. 다음에는 Python 관련 응용 프로그램을 설치해보는 것도 재밌을 것 같다.

<6주차>



➤ 설치 프로그램:

Jupyter Notebook(<https://jupyter.org/>)

➤ 프로그램 설명:

Jupyter Notebook은 코드, 수식, 시각화, 주석 등을 모두 담은 문서를 만들고 공유할 수 있는 웹기반의 오픈소스 소프트웨어이다. 데이터 변환, 통계 모델링, 머신 러닝 등 여러 분야에서 사용이 되고 있다. 특히 HTML, 이미지, 비디오 등 다양한 output을 제공하는 것이 큰 특징이다.

(출처: <https://jupyter.org/>)

➤ 설치 이유:

저번주에 C++ 개발환경을 구축하기 위해 Visual Studio Code를 설치했었다. 물론 Python도 Visual Studio Code에서 확장팩을 통해서 이용할 수 있다. 하지만 Python은 주로 데이터 분석 등을 할 때 사용해왔었기 때문에 좀 더 그 분야에 어울리는 환경을 구축하고 싶었다. 그 중에 Jupyter Notebook을 전에 이용해보기도 했었고, 꽤나 많은 사람들이 이용하는 소프트웨어이기 때문에 설치하기로 했다. 또한 이번 학기에 듣는 기계학습 수업에서도 Jupyter Notebook을 사용하기 때문에 여러모로 필요한 프로그램이라고 생각했다.

➤ 설치 과정:

Code Project라는 코딩 커뮤니티에 Jupyter Notebook을 설치하는 법을 친절하게 알려주는 글(<https://www.codeproject.com/Articles/5279078/Install-Jupyter-Notebook-on-Ubuntu>)이 있어서 이 글을 참고하였다. 소개된 명령어들을 마찬가지로 한 줄씩 살펴보았다.

```
sudo apt-get update && sudo apt-get upgrade
```

이 두 줄은 모두 패키지 관리 툴인 apt-get을 이용한 명령어임을 알 수 있었다. 두 줄 모두 전에 본 적이 있다. Update 명령어는 설치 가능한 패키지 리스트를 업데이트하는 것이고, upgrade 명령어는 실제 설치한 패키지들을 최신 버전으로 업그레이드 하는 명령어이다. 실제 패키지를 설치하기 전에 준비하는 단계로 볼 수 있다.

```
sudo apt install python3-pip python3-dev
```

이 줄도 패키지 관리 툴인 apt를 이용한 명령어이다. 다만 뒤의 두 패키지의 정체는 알지 못했다. 알아보니 우선 python3-pip은 python에서 패키지를 관리하고 설치하는 도구라고 한다. Python3-dev는 개발자 패키지로, 컴파일에 필요한 헤더, 라이브러리 등이 들어가 있다고 한다. 즉 pip이라는 패키지 관리자를 통해 Python3-dev라는 이름의 개발자 패키지를 설치하라는 의미인 것이다.

```
sudo -H pip3 install --upgrade pip  
sudo -H pip3 install virtualenv
```

우선 첫 줄은 설치한 pip 관리자를 최신 버전으로 업그레이드하라는 의미인 것 같다. 두번째 줄의 virtualenv가 무엇인지 궁금하여 찾아보았다. 찾아보니 python의 기능을 선택적으로 쓸 수 있는 가상환경 모듈이라고 한다. Jupyter notebook의 이용을 위해서는 python 가상환경을 구축해야 하는데, 이때 쓰일 것 같다.

```
mkdir code
```

이 줄은 설명과 함께 보니 직관적으로 make directory의 약자로, code라는 이름의 새 디렉토리를 만드라는 의미인 것을 쉽게 알 수 있었다.

```
cd code  
virtualenv jupyterenvironment
```

먼저 첫째줄로 디렉토리를 code로 이동하는 것을 볼 수 있다. 아랫줄은 의미를 봤을 때 좀 전에 설치한 virtualenv 모듈을 이용하여 jupyter 가상환경을 구축하는 줄이라고 할 수 있다.

```
pip install jupyter
```

마지막 줄은 비교적 간단했다. 가상환경이 구축되었으면, 그 위에 pip 패키지 관리자로 jupyter notebook을 설치하는 것이다. 터미널에서 모두 실행하니 jupyter notebook을 성공적으로 설치할 수 있었다.

```
jhhhan0208@jhhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt-get update && sudo apt-get upgrade  
[sudo] jhhhan0208의 암호:
```

```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install python3-pip python3-dev
```

```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo -H pip3 install --upgrade pip
Collecting pip
```

```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo -H pip3 install virtualenv
Collecting virtualenv
```

```
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ mkdir code
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ cd code
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/code$ virtualenv jupyterenvironment
created virtual environment CPython3.8.5.final.0-64 in 211ms
creator CPython3Posix(dest=/home/jhhan0208/code/jupyterenvironment, clear=False,
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundl
added seed packages: pip==21.0.1, setuptools==54.1.2, wheel==0.36.2
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,Py
jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/code$ source jupyterenvironment/bin/
```

```
(jupyterenvironment) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/code$ pip install jupyter
Collecting jupyter
```

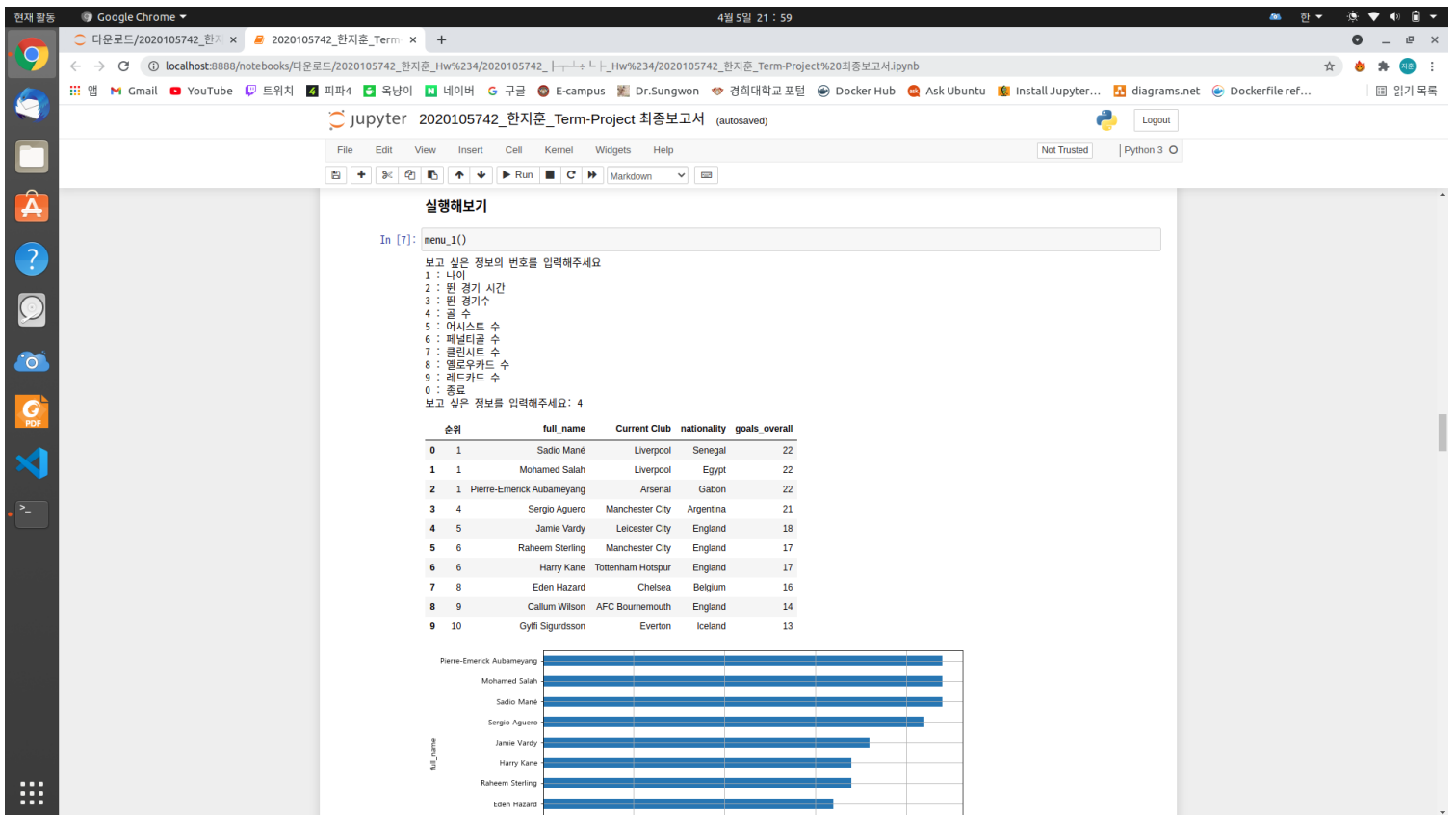
➤ 사용/느낀 점:

우선 설치 후에는 콘솔에 jupyter notebook이라고 쳐서 바로 실행할 수 있었다.

```
(jupyterenvironment) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/code$ jupyter notebook
[I 22:53:06.551 NotebookApp] Writing notebook server cookie secret to /home/jhhan0208/.
[I 22:53:06.719 NotebookApp] Serving notebooks from local directory: /home/jhhan0208/co
[I 22:53:06.719 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 22:53:06.719 NotebookApp] http://localhost:8888/?token=c72c8a512c5a062774ea07da33952
[I 22:53:06.719 NotebookApp] or http://127.0.0.1:8888/?token=c72c8a512c5a062774ea07da3
[I 22:53:06.719 NotebookApp] Use Control-C to stop this server and shut down all kernel
[C 22:53:06.736 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/jhhan0208/.local/share/jupyter/runtime/nbserver-17836-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=c72c8a512c5a062774ea07da33952ecb14b1452e262b2124
or http://127.0.0.1:8888/?token=c72c8a512c5a062774ea07da33952ecb14b1452e262b2124
```

실행을 하고 이전에 진행했었던 Term-Project도 실행해보았다.



이제 이후에 term 프로젝트나 기타 데이터 분석을 할 때 우분투에서 하면 되겠다고 생각했다. 이전에 윈도우에서도 jupyter notebook을 사용했는데 차이점은 편리함의 정도인 것 같다. 우선 터미널에 jupyter notebook만 치면 쉽게 실행이 가능하다. 또한 아래처럼 server를 닫을 때도 ctrl + c를 두 번 돌려주면 된다. 그리고 anaconda, pillow 등도 설치해보았는데 터미널을 통해 설치하니까 정말 간단했다.

벌써 리눅스로 한학기 살기 프로젝트를 반 넘게 진행하였다. 매주를 거듭하면서 점차 리눅스의 명령어들, 프로그램들에 대한 이해를 조금씩 더 하게 되는 느낌이다. 이 프로젝트가 끝날 때가 되면 그래도 누가 리눅스 우분투에 대해 물어봤을 때 당당하게 대답할 수 있는 사람이 되고 싶다. 남은 주에도 개발에 도움이 되는 다른 여러 프로그램들을 설치해 나갈 계획이다.

<7주차>



➤ 설치 프로그램:

LibreOffice Calc(<https://ko.libreoffice.org/discover/calc/>)

➤ 프로그램 설명:

Calc는 LibreOffice에서 제공하는 spreadsheet 프로그램이다. 간단한 UI 덕분에 초심자들에게 적합하고, 고급 기능들을 여러 개 포함하고 있어 전문가에게도 적합하다. 동시에 여러 사람들이 문서를 열람할 수 있게 해주며, 다른 사람과 파일을 공유하고 수정하는 것이 가능하다. 또한 MS 오피스 엑셀 등 다른 spreadsheet 프로그램들과의 호환성도 우수한 편이다.

➤ 설치 이유:

데이터 분석 관련 프로젝트들을 해보니 Raw 데이터의 중요성에 대해서 느끼게 되었다. 다양한 소스에서 Raw 데이터를 추출하고, python의 Numpy등의 도구로 그 데이터를 내 필요에 맞게 다듬는 과정이 필수적이라고 느꼈다. 대부분의 데이터는 Excel 등의 스프레드시트 형태로 존재하는데, 현재 나의 Ubuntu에는 스프레드시트를 보고 편집할 수 있는 프로그램이 없었다. 이런 프로그램에는 무엇이 있는지 찾아보았고, 그 중에 사용자도 많고 데이터 분석 도구화 호환도 되는 Calc를 설치하기로 하였다.

➤ 설치 과정:

이번주는 Linuxhint의 설명글(<https://linuxhint.com/install-libreoffice-ubuntu-linux-mint/>)을 참고하여 설치를 진행하였다.

```
$ sudo apt update
```

이제 이 명령어는 너무 자주 봐서 익숙하다. 설치 패키지들의 업데이트가 있는지 확인하는 명령어이다. 설치 전에 준비를 하는 느낌이라고 생각한다.

```
$ sudo apt install libreoffice
```

Libreoffice의 설치 과정은 다른 소프트웨어에 비해서 훨씬 간단하였다. 패키지 관리 툴을 이용하여 관리자 권한으로 설치하면 설치가 끝난다고 한다. 듣기로는 Ubuntu 최초 설치시에 기본 프로그램을 같이 설치한다고 하면 자동으로 설치될 정도로 기초적이고 많은 사람들이 사용하는 프로그램이라서 그런듯 하다.

바로 터미널을 통해 설치를 진행하였다.

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt update
기존:1 https://download.docker.com/linux/ubuntu focal InRelease
기존:2 http://dl.google.com/linux/chrome/deb stable InRelease
기존:3 http://packages.microsoft.com/repos/code stable InRelease
기존:4 https://packages.microsoft.com/repos/vscode stable InRelease
기존:5 http://ppa.launchpad.net/libreoffice/libreoffice-7-0/ubuntu focal InRelease
받기:6 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
기존:7 http://kr.archive.ubuntu.com/ubuntu focal InRelease
받기:8 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
기존:9 http://ppa.launchpad.net/libreoffice/ppa/ubuntu focal InRelease
받기:10 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
내려받기 324 k바이트, 소요시간 2초 (146 k바이트/초)
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
모든 패키지가 최신입니다.
```

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install libreoffice
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
```

```
Creating config file /etc/libreoffice/registry/calc.xcd with new version
libreoffice-nlpsolver (0.9+Lib07.1.2-rc2-0ubuntu0.20.04.1~lo3) 설정하는 중입니다 ...
libreoffice (1:7.1.2-rc2-0ubuntu0.20.04.1~lo3) 설정하는 중입니다 ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libreoffice-common (1:7.1.2-rc2-0ubuntu0.20.04.1~lo3) ...
Processing triggers for fontconfig (2.13.1-2ubuntu3) ...
```

설치 후에는 다음 명령어를 통해 소프트웨어의 버전도 확인할 수 있었다.

```
$ libreoffice --version
```

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ libreoffice --version
LibreOffice 7.1.2.2 10(Build:2)
```

➤ 사용/느낀 점:

설치를 완료하니 내가 원했던 Calc 말고도 Base, Draw 등 Libre Office의 여러 응용 프로그램이 함께 설치된 것을 볼 수 있다.



Calc를 실행해보니 전반적으로 깔끔한 느낌이었고, Excel과 비슷하다고도 생각했다. 프로젝트에 사용했던 Raw data 파일을 Excel과 Calc에서 모두 열어서 비교해보았다.

england-premier-league-players-2018-to-2019-stats									
full_name									
	A	B	C	D	E	F	G	H	I
1	full_name	age	birthday	league	season	position	Current Club	minutes_playe	minutes_playe
2	Aaron Cress	31	629683200	Premier Leagu	2018/2019	Defender	West Ham Unit	1589	888
3	Aaron Lennc	33	545526000	Premier Leagu	2018/2019	Midfielder	Burnley	1217	487
4	Aaron Mooy	30	653353200	Premier Leagu	2018/2019	Midfielder	Huddersfield Tc	2327	1190
5	Aaron Rams	30	662169600	Premier Leagu	2018/2019	Midfielder	Arsenal	1327	689
6	Aaron Rowe	20	968281200	Premier Leagu	2018/2019	Forward	Huddersfield Tc	69	14
7	Aaron Wan-	23	880502400	Premier Leagu	2018/2019	Midfielder	Crystal Palace	3135	1605
8	Abdelhamid	24	849139200	Premier Leagu	2018/2019	Midfielder	Huddersfield Tc	49	0
9	Abdoulaye D	28	725846400	Premier Leagu	2018/2019	Midfielder	Watford	3062	1566
10	Aboubakar k	26	794534400	Premier Leagu	2018/2019	Forward	Fulham	687	468
11	Adalberto Pe	23	865033200	Premier Leagu	2018/2019	Forward	Watford	0	0
12	Adam David	32	579222000	Premier Leagu	2018/2019	Midfielder	Liverpool	465	189
13	Adam Masir	27	757468800	Premier Leagu	2018/2019	Defender	Watford	1003	463
14	Adam Smith	29	672879600	Premier Leagu	2018/2019	Defender	AFC Bournemou	2073	1051
15	Adama Diak	24	826521200	Premier Leagu	2018/2019	Forward	Huddersfield Tc	551	245

england-premier-league-players-2018-to-2019-stats.csv - LibreOffice Calc

파일(F) 편집(E) 보기(V) 삽입(I) 서식(O) 스타일(Y) 시트(S) 데이터(D) 도구(T) 창(W) 도움말(H)

10 pt B I U A

A1 f. Σ = full_name

	A	B	C	D	E	F	G	H	I
1	full_name	age	birthday	league	season	position	Current Club	minutes_played_overall	minutes_played_home
2	Aaron Cresswell	31	629683200	Premier League	2018/2019	Defender	West Ham United	1589	888
3	Aaron Lennon	33	545526000	Premier League	2018/2019	Midfielder	Burnley	1217	487
4	Aaron Mooy	30	653353200	Premier League	2018/2019	Midfielder	Huddersfield Town	2327	1190
5	Aaron Ramsey	30	662169600	Premier League	2018/2019	Midfielder	Arsenal	1327	689
6	Aaron Rowe	20	968281200	Premier League	2018/2019	Forward	Huddersfield Town	69	14
7	Aaron Wan-Bissaka	23	880502400	Premier League	2018/2019	Midfielder	Crystal Palace	3135	1605
8	Abdelhamid Sabiri	24	849139200	Premier League	2018/2019	Midfielder	Huddersfield Town	49	0
9	Abdoulaye Doucoure	28	725846400	Premier League	2018/2019	Midfielder	Watford	3062	1566
10	Aboubakar Kamara	26	794534400	Premier League	2018/2019	Forward	Fulham	687	468
11	Adalberto Peñaranda Maestre	23	865033200	Premier League	2018/2019	Forward	Watford	0	0
12	Adam David Lallana	32	579222000	Premier League	2018/2019	Midfielder	Liverpool	465	189
13	Adam Masina	27	757468800	Premier League	2018/2019	Defender	Watford	1003	463
14	Adam Smith	29	672879600	Premier League	2018/2019	Defender	AFC Bournemouth	2073	1051
15	Adama Diakhaby	24	836521200	Premier League	2018/2019	Forward	Huddersfield Town	551	345
16	Adama Traoré Diarra	25	822528000	Premier League	2018/2019	Midfielder	Wolverhampton Wanderers	890	315
17	Ademola Lookman	23	877302000	Premier League	2018/2019	Forward	Everton	601	334
18	Adrian Mariappa	34	528678000	Premier League	2018/2019	Defender	Watford	1921	841
19	Adrian San Miguel del Castillo	34	536630400	Premier League	2018/2019	Goalkeeper	West Ham United	0	0
20	Adrien Sebastian Perruchet Silva	32	605923200	Premier League	2018/2019	Midfielder	Leicester City	88	8
21	Ainsley Maitland-Niles	23	872809200	Premier League	2018/2019	Midfielder	Arsenal	985	462
22	Alberto Moreno	28	710290800	Premier League	2018/2019	Defender	Liverpool	155	90
23	Aleksandar Mitrovic	26	779670000	Premier League	2018/2019	Forward	Fulham	3282	1616
24	Alex Iwobi	24	831078000	Premier League	2018/2019	Forward	Arsenal	1972	742
25	Alex McCarthy	31	628646400	Premier League	2018/2019	Goalkeeper	Southampton	2250	1170
26	Alex Oxlade-Chamberlain	27	745369200	Premier League	2018/2019	Midfielder	Liverpool	19	19
27	Alex Pritchard	27	736383600	Premier League	2018/2019	Midfielder	Huddersfield Town	2092	992
28	Alex Smithies	31	636595200	Premier League	2018/2019	Goalkeeper	Cardiff City	0	0
29	Alexander Sørloth	25	818121600	Premier League	2018/2019	Forward	Crystal Palace	173	121
30	Alexandre Lacazette	29	675385200	Premier League	2018/2019	Forward	Arsenal	2503	1403
31	Alexandre Nascimento Costa Silva	24	858470400	Premier League	2018/2019	Forward	West Ham United	17	0
32	Alexis Sanchez	32	598492800	Premier League	2018/2019	Forward	Manchester United	877	387
33	Alfie Jones	23	876178800	Premier League	2018/2019	Defender	Southampton	0	0

이전에 `read_csv()` 등의 함수를 통해 엑셀 파일을 Jupyter Notebook으로 가져와서 데이터 분석 라이브러리로 분석한 적이 있다. Calc 파일의 경우에도 Unotools(<https://pypi.org/project/unotools/>)라는 응용 프로그램으로 비슷하게 가져올 수 있다고 한다. 나중에 우분투와 더욱 친해질 겸 Calc 파일을 바탕으로도 분석 프로젝트를 진행해볼 생각이다.

또한 의도와는 다르게 Libre Office의 다른 프로그램들도 같이 설치되었는데, 그 프로그램들도 나중에 한번 사용해볼 생각이다. 확실히 Libre Office는 MS Office와 비슷한 느낌이 많이 든다. Ubuntu가 지원하는 MS Office를 대신할 만한 소프트웨어를 성공적으로 설치하고 사용할 수 있게 되어 뿌듯했다.

<8주차>

➤ 설치 프로그램:

Github Desktop(<https://desktop.github.com/>)

➤ 프로그램 설명:

Github Desktop는 Git를 더 편리하게 사용할 수 있는 응용 프로그램이다. Github의 repository 들을 가져와서 기본적인 commit, pull request 등의 기능을 프로그램 상에서 할 수 있다. 추가로 소스 코드파일을 언어에 맞게 editor에서 바로 열어볼 수 있다. 완전한 open source로, 많은 사용자들이 Github Desktop을 통해 협업을 진행하고 있다.

(출처: <https://desktop.github.com/>)

➤ 설치 이유:

8주차 수업에서 software maintenance에 대해 배우면서 해당 분야의 핵심인 Git, Github에 대해 배웠다. 작년 방학때도 과제로 Github상에 나의 portfolio 홈페이지를 만들었기 때문에 Git을 접할 일이 많았다. 내 Portfolio를 관리하고 Git에 대해 더 알아볼 겸 해서 관련 프로그램에는 무엇이 있는지 알아보았다. 찾아보던 중 Github의 기능들을 좀 더 편한 GUI상에서 사용할 수 있는 Github Desktop이 눈에 들어와서 설치하여 사용해보기로 하였다.

➤ 설치 과정:

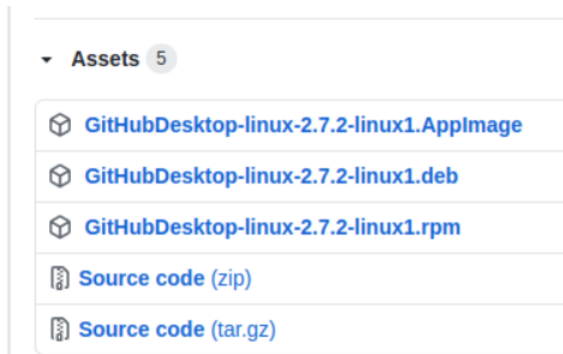
이번주는 MeshWorld의 Github Desktop 설치 tutorial

(<https://meshworld.in/install-github-desktop-on-ubuntu-20-04-or-ubuntu-based-distributions/>)

을 참고하였다.

우선 Github Desktop은 윈도우와 맥은 공식적으로 지원을 하지만, Linux는 공식 지원을 하지는 않는다고 한다. 다만 참고한 글을 보니 Brendan Forster 개발자 분이 Github repository 형태로 Linux에서도 사용할 수 있게 했다고 한다. 우선 지시에 따라 해당repository

(<https://github.com/shiftkey/desktop/releases>)에서 .deb 파일을 다운받았다. 3주차에 .deb이 소프트웨어 패키지 포맷의 확장자라는 것이 기억났다. Wget 등으로 받을 수도 있었지만, Git 페이지도 보기 위해 지시대로 사이트에서 받았다. 여러 프로젝트를 진행하면서 느끼는 거지만 정말 Github에는 모든 것이 있는 것 같다.



```
sudo dpkg -i fileName.deb
```

첫 줄에 dpkg라는 처음보는 명령어가 등장하였다. 찾아보니 dpkg는 debian package의 약자로, debian 즉 .deb 확장자의 파일을 설치하고 삭제하는 데에 사용하는 소프트웨어라고 한다. 보통 Ubuntu Software Center을 거치지 않고 .deb 파일을 직접 설치하고 싶을 때 사용한다고 한다.

```
sudo apt update && sudo apt install -f
```

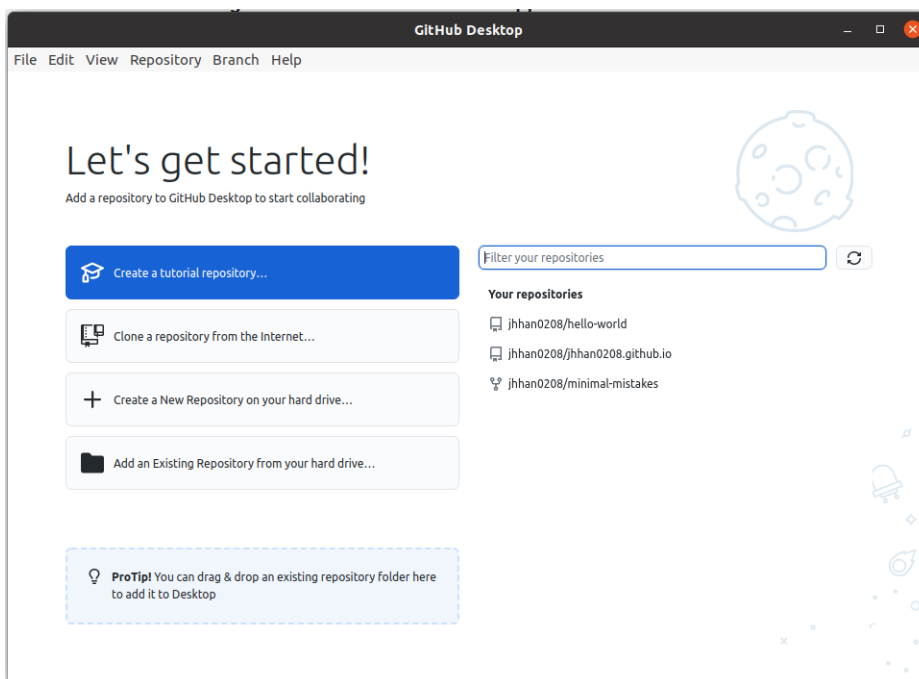
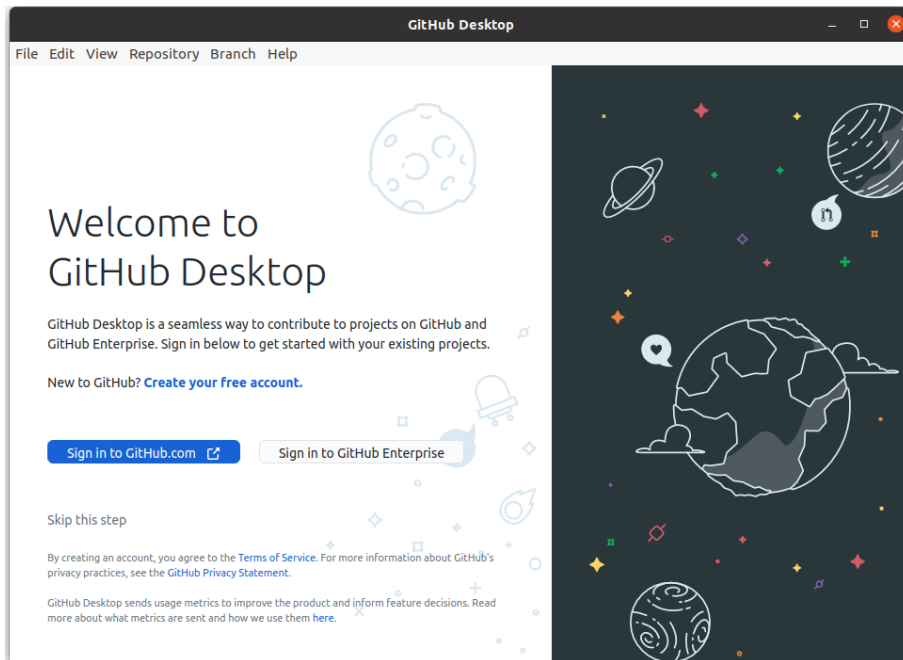
이 줄은 더의 단골로 들어가는, 패키지들의 업데이트 상태를 살피는 줄들이다. 다만 여기서는 받은 패키지에 오류가 있는지 등을 보기 위해 패키지 설치 이후에 한번 돌려준다고 하였다. 이 줄 실행 후 dpkg를 통해 재설치를 진행하면, 안정적으로 Github Desktop을 설치할 수 있다고 한다.

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/다운로드$ sudo dpkg -i GitHubDesktop-linux-2.7.2-linux1.deb
[sudo] jhhan0208의 암호:
Selecting previously unselected package github-desktop.
(데이터베이스 읽는중 ...현재 314957개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack GitHubDesktop-linux-2.7.2-linux1.deb ...
Unpacking github-desktop (2.7.2-linux1) ...
github-desktop (2.7.2-linux1) 설정하는 중입니다 ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/다운로드$ sudo apt update && sudo apt install -f
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~/다운로드$ sudo dpkg -i GitHubDesktop-linux-2.7.2-linux1.deb
(데이터베이스 읽는중 ...현재 316646개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack GitHubDesktop-linux-2.7.2-linux1.deb ...
Unpacking github-desktop (2.7.2-linux1) over (2.7.2-linux1) ...
github-desktop (2.7.2-linux1) 설정하는 중입니다 ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
```

터미널에서 실행하니 성공적으로 설치가 된 것을 볼 수 있었다.

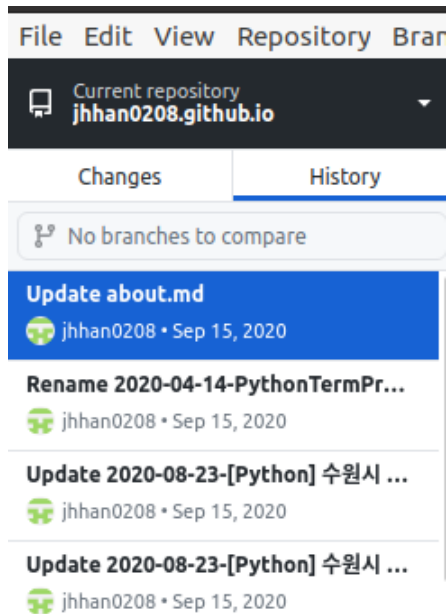
➤ 사용/느낀 점:

Github Desktop을 실행해보니 친절한 UI와 함께 Github과 연동시키는 페이지가 나오고, 이후에는 연결된 repository들을 보여주었다.

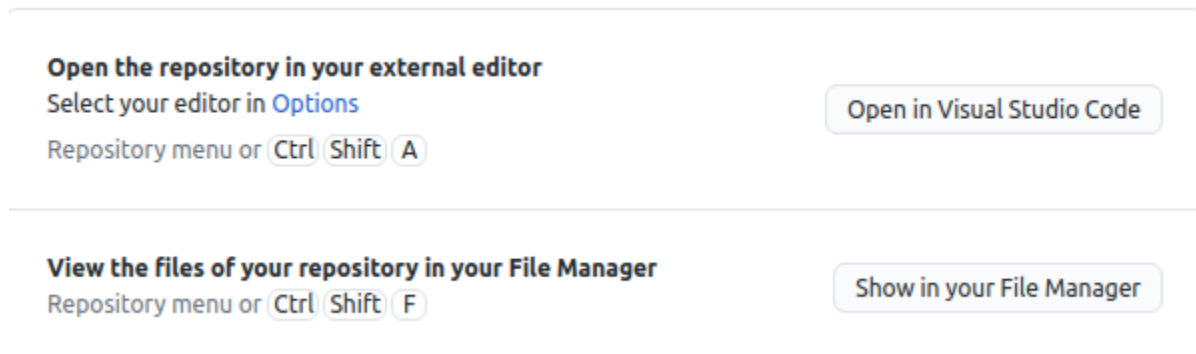


그 중에 나의 포트폴리오 repository인 jhhan0208.github.io를 열어보았다. 대강 기능들을 둘러봤는데, 포트폴리오를 관리하기에 상당히 괜찮은 기능들이라는 것을 느꼈다. 우선 History 탭

에 날짜별로 repository에 대한 수정 내용이 정리되어 있어 어떤 과정으로 페이지를 만들어왔는지 알기 용이하였다.



또한 처음에 추천해주는 기능들도 사용해보았다.

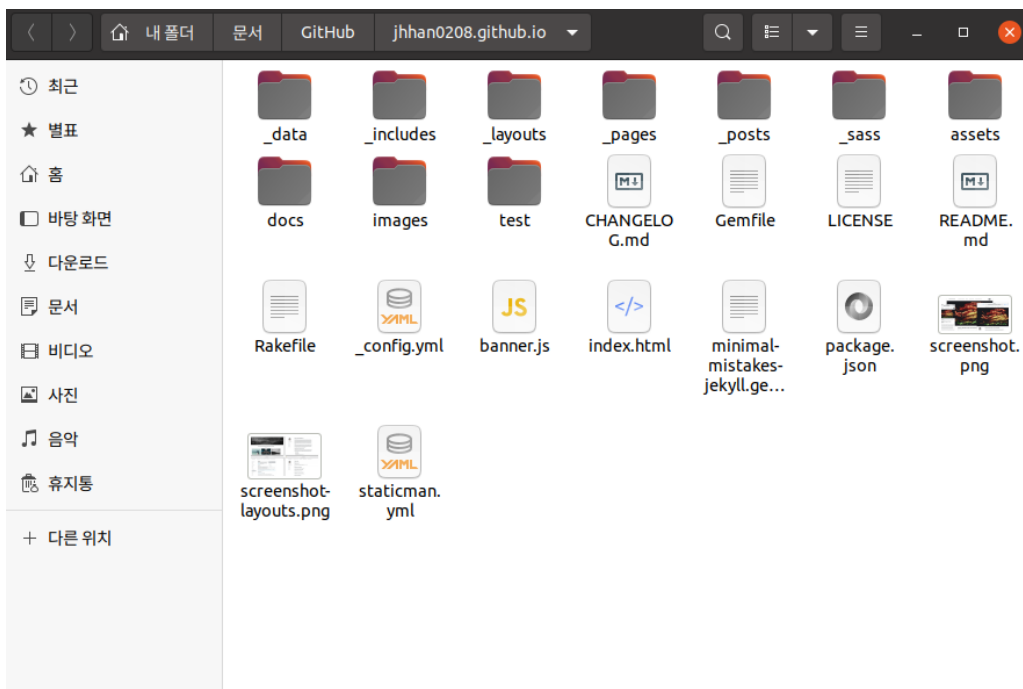


우선 선택한 editor로 바로 repository를 볼 수 있는 기능이 아주 마음에 들었다. 5주차에 visual studio code를 설치해 봤기 때문에 바로 열어볼 수 있었다.(이렇게 도움이 될 줄은 몰랐다.)

The image shows a VS Code editor window with the Explorer sidebar on the left and the Editor pane on the right. The Explorer sidebar shows a file tree for a project named 'JHHAN0208.GITHUB.IO'. The file tree includes folders like '_data', '_includes', '_layouts', '_pages', '_posts', '_sass', '.github', 'assets', 'docs', 'images', and 'test'. It also lists files like '_config.yml' (highlighted), '.editorconfig', '.gitattributes', '.gitignore', '.travis.yml', 'banner.js', 'CHANGELOG.md', 'Gemfile', 'index.html', and 'LICENSE'. The Editor pane shows the content of '_config.yml', which is a Jekyll configuration file. The file contains comments and settings for theme, remote theme, site settings, and site title. The settings are as follows:

```
1 # Welcome to Jekyll!
2 #
3 # This config file is meant for settings that affect your entire
4 # which you are expected to set up once and rarely need to edit
5 # For technical reasons, this file is "NOT" reloaded automatica
6 # 'jekyll serve'. If you change this file, please restart the s
7
8 # Theme Settings
9 #
10 # Review documentation to determine if you should use 'theme' or
11 # https://mmistakes.github.io/minimal-mistakes/docs/quick-start
12
13 # theme : "minimal-mistakes-jekyll"
14 # remote_theme : "mmistakes/minimal-mistakes"
15 minimal_mistakes_skin : "default" # "default", "air", "aqua"
16
17 # Site Settings
18 locale : "en-US"
19 title : "Han Jihoon"
20 title_separator : "-"
21 subtitle : # site tagline that appears below site title
22 name : "Han Jihoon"
23 description : "My Python Data Science Portfolio"
24 url : # the base hostname & protocol for your site
```

사실 작년 방학 때 처음 홈페이지를 만들 때는 github 페이지 자체에서 portfolio를 수정하는 경우가 많았는데, editor에서 보는 것에 비해 훨씬 가독성이 좋지 않았다. 또한 atom 등의 다른 editor을 별도로 번갈아 쓰는 것도 어려웠는데, 이렇게 편한 프로그램을 너무 늦게 찾았다는 생각이 들었다. 연결 파일 디렉토리를 바로 볼 수 있는 두 번째 기능도 편리했다.



페이지에 첨부 사진, 자료 등을 넣을 때 용이하게 사용할 수 있을 것이다.

이번 주차에 software 유지보수의 관점에서 Git에 대해 살펴보았는데, 관련 프로그램인 Github Desktop을 직접 설치해보고 나의 portfolio 관리 목적으로 사용할 수 있게 되어서 뿌듯했다. 지금은 개인 홈페이지를 관리하는 수준이지만, 앞으로 더 복잡한 프로그램, 소프트웨어들을 만든 후에 그것들을 관리하는 데에 있어서 Github Desktop 등의 software maintenance tool들이 큰 도움을 줄 것 같다. 이번 학기가 끝나면 군대로 가는데, 가기 전에 이번 학기 프로젝트들을 추가로 사이트에 올리고 정리라도 조금 해놓고 가야겠다고 생각했다.

<9주차>



➤ 설치 프로그램:

Slack(<https://slack.com/intl/ko-kr/>)

➤ 프로그램 설명:

Slack는 클라우드 기반의 팀 협업 도구이다. 커뮤니케이션에 필요한 대화, 전화, 파일 공유 등 다양한 방법이 지원된다. 메시지의 친숙한 형식을 사용하기 때문에 사용하기에도 편리한 프로그램이다.

➤ 설치 이유:

해당 수업에서 협업을 위한 소프트웨어와 방법에 대해서 배웠다. 수업 중 협업에 도움이 되는 소프트웨어 부분을 들었을 때, 저번 학기에 진행했던 디자인적 사고 협업 프로젝트 등에 사용했으면 좋았겠다고 생각하면서 관심을 가지게 되었다. 소개된 여러 개의 소프트웨어 중에 Slack을 설치하여 직접 사용해보고, 이후 진행할 팀 프로젝트에 어떤 식으로 활용할 수 있을지를 고민해보기로 하였다.

➤ 설치 과정:

프로젝트를 진행함에 따라 점점 무조건적으로 설명 글을 참고하지는 않게 되었다. 설치하려는 Slack의 snap package가 있을 것이라고 생각했고, 찾아보니 존재했다. 그러니 여러 번 했던 것처럼 snap 패키지 관리자를 통해서 바로 설치를 진행할 수 있었다.

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo snap install slack --classic
2021/05/10 00:26:23.968744 main.go:176: description of download's "channel" is lowercase: "stable 대신 이 채널을 사
용합니다"
2021/05/10 00:26:23.968837 main.go:176: description of download's "revision" is lowercase: "snap 의 주어진 리비전을
다운로드 합니다, developer access 가 필요합니다"
```

```
snap "slack"의 데이터 복사
slack4.15.0 from Slack✓ installed
```

➤ 사용/느낀 점:

우선 설치를 완료하니 웹에서 로그인하라는 창이 떴다. 로그인을 마치고 다시 돌아왔다.



돌아오니 팀의 이름을 입력하라고 하였다. 지금은 전반적인 기능들만 살펴보기 위해 임시로 Term-Project라고 지었다. 나중에는 진짜 프로젝트로 짓게 될 것이다.

1/3단계

회사 또는 팀 이름이 어떻게 됩니까?

Slack 워크스페이스의 이름이 됩니다. 팀이 인식할 수 있는 이름을 입력하세요.

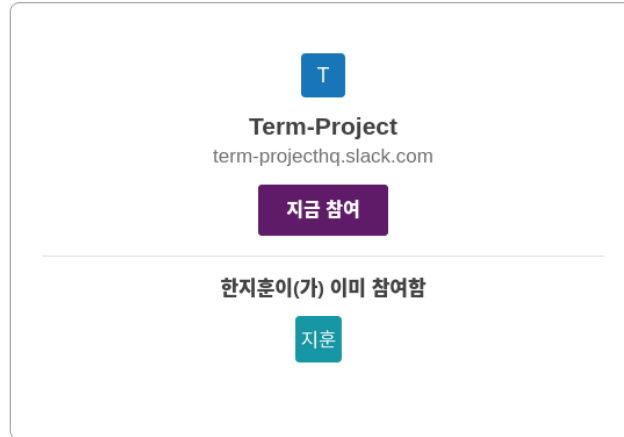
Term-Project

38

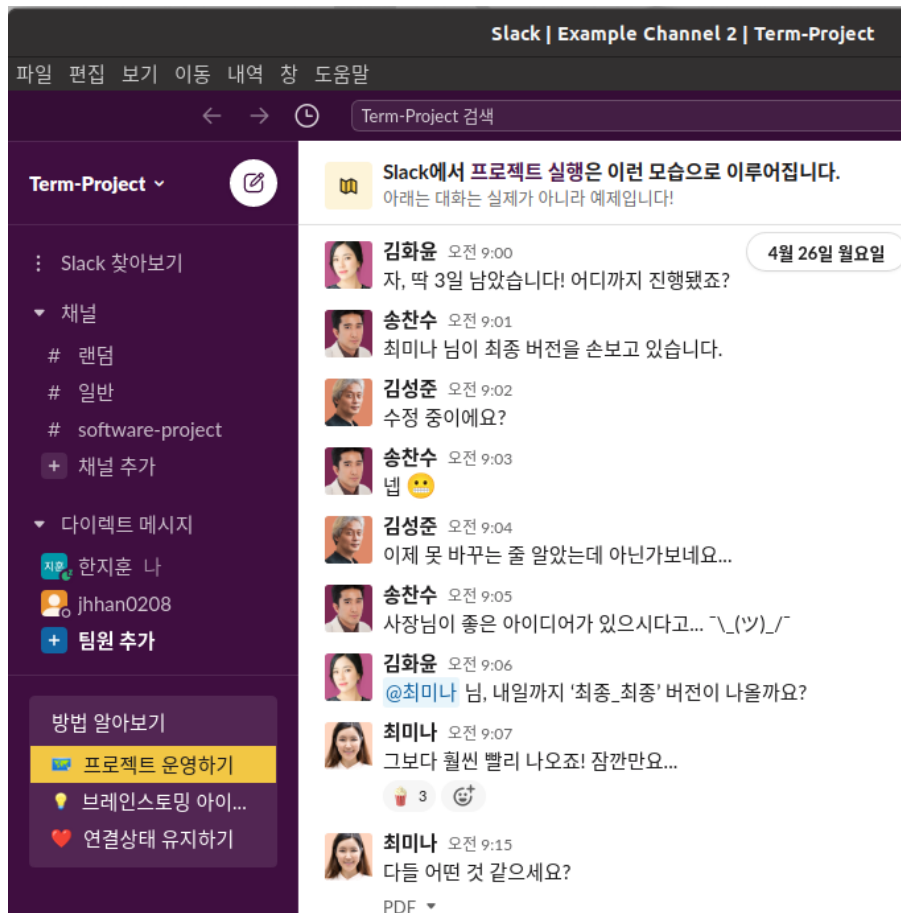
이메일을 통해서 팀원들을 편하게 초대할 수 있었다. 지금은 나의 경희대 메일 계정을 초대해보았다.

Slack에서 팀 참여

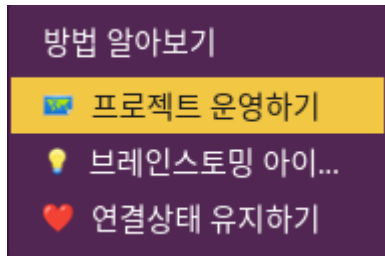
한지훈 (jhhan0208@gmail.com) 님이 고객님의 **Term-Project**(아)라는 워크스페이스에서 Slack을 함께 사용하도록 초대했습니다.



들어가보니 전반적인 느낌은 카카오톡 등의 메신저와 비슷한 형태였다. 그러나 pdf 등 파일을 공유하기가 더 쉽고, 프로젝트 관리에 좀 더 특화된 느낌이 있었다.



Tutorial에서 크게 3가지 기능을 소개했다. 예시를 통해 친절하게 설명이 되어있었다. 나중에 어떤 특성을 중심으로 이 프로그램을 사용할지 알기 위해서 예시에서 보여주는 기능들을 요약해보았다.



1. 프로젝트 운영

직관적으로 볼 수 있듯이 대화방에서 메신저의 형태로 회의를 진행하는 것이 가능하다. 이 밖에도 통화 기능을 통한 단체 대화방 생성 등의 기능도 있다.

2. 브레인스토밍

팀원들은 대화방에서 글, 사진, 문서파일 등 다양한 형태로 자신의 아이디어를 제시하는 것이 가능하다. 또한 제시한 아이디어들에 대해 SNS의 '좋아요' 형태로 찬반 피드백을 줄 수 있고, 추가적인 보완점 등도 제시하기 편리하다.

3. 연결상태 유지

번역이 살짝 어색한 것 같은데, 내용을 보니 원만한 관계를 유지하는 내용 정도로 볼 수 있을 것 같다. 회의 등 일만 하는 것이 아니라, 쉴 때 무엇을 하는지 등 일상을 공유하면서 편하게 대화하는 내용이 소개되었다. UI 자체가 SNS와 비슷하다 보니 이런 친목의 기능도 할 수 있는 것 같다.

Slack을 직접 설치해보고 전반적인 기능들을 살펴보았다. 협업에 있어서 SW가 줄 수 있는 이점에 대해서는 생각해보지 못했는데, SW의 역할을 너무 과소평가한 것 같다. 그리고 한편으로는 이전의 팀 프로젝트에서 이런 종류의 프로그램을 사용하지 않은 것이 아쉽기도 했다. 이후에 진행하는 프로젝트에서는 Slack를 꼭 사용해보기로 결심했다.

<10주차>

➤ 설치 프로그램:

SnakeViz(<https://jiffyclub.github.io/snakeviz/>)

➤ 프로그램 설명:

SnakeViz는 python의 cProfile 모듈을 볼 수 있는 브라우저 기반의 그래픽 뷰어이다. cProfile은 python에서 프로그램의 여러 부분이 얼마나 자주 그리고 얼마나 오랫동안 실행되었는지를 알려주는 통계이다. 이 통계를 브라우저에서 바로 볼 수 있는 것이다.

➤ 설치 이유:

수업시간에 프로그래밍에 있어 Profiler가 어떤 역할을 하는지, 그리고 예시들을 살펴보는 부분이 있었다. 그동안 python으로 여러 프로젝트들을 진행했지만 프로젝트의 성능을 객관적으로 보기가 어려웠다. 그냥 좀 오래 돌아가면 성능이 조금 아쉽다 정도로만 생각했었다. 그래서 수업 때 배운 python의 profiler중 하나인 SnakeViz를 직접 설치하여 사용해보기로 하였다.

➤ 설치 과정:

SnakeViz의 Github 페이지(<https://jiffyclub.github.io/snakeviz/>)를 살펴보니 pip 패키지로 설치하는 것을 권장하였다. 따라서 해당 방법으로 설치를 진행하였다.

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ pip install snakeviz
Collecting snakeviz
  Downloading snakeviz-2.1.0-py2.py3-none-any.whl (282 kB)
    | 282 kB 877 kB/s
Requirement already satisfied: tornado>=2.0 in ./env/lib/python3.8/site-packages (from snakeviz) (6.0.4)
Installing collected packages: snakeviz
Successfully installed snakeviz-2.1.0
```

➤ 사용/느낀 점:

설치한 SnakeViz를 현재 진행 중인 data project 에 적용해보기로 하였다. 현재로서는 원본 data를

불러와서 preprocessing하는 단계까지 왔는데, preprocessing 과정이 어떻게 진행되는지를 시각화해보았다. 우선 외부 확장자로 SnakeViz를 Jupyter Notebook로 불러왔다.

```
In [1]: %load_ext snakeviz
```

원본 파일 불러오기

```
In [2]: import pandas as pd

data = pd.read_csv('data.csv')
#data.head(20)
```

다음으로 preprocessing 단계를 하나의 함수 안에 넣고 SnakeViz로 그 함수의 돌아가는 과정을 보기로 했다.

1차 업데이트 + 2차 업데이트 + 3차 업데이트

```
In [18]: def Update_combine():
    first_list_combine = first_list.sort_values(by = 'name', ascending = True)
    second_list_combine = second_list.sort_values(by = 'name', ascending = True)
    third_list_combine = third_list.sort_values(by = 'name', ascending = True)

    first_list_combine = first_list_combine.reset_index(drop=True)
    second_list_combine = second_list_combine.reset_index(drop=True)
    third_list_combine = third_list_combine.reset_index(drop=True)

    second_dif = []
    third_ovr = []

    third_dif = []
    fourth_ovr = []

    for i in range(0, 397):
        second_dif.append(int(second_list_combine['third_ovr'][i]) - int(first_list_combine['second_ovr'][i]))
        third_ovr.append(int(second_list_combine['third_ovr'][i]))
        third_dif.append(int(third_list_combine['fourth_ovr'][i]) - int(second_list_combine['third_ovr'][i]))
        fourth_ovr.append(int(third_list_combine['fourth_ovr'][i]))

    first_list_combine['second_dif'] = second_dif
    first_list_combine['third_ovr'] = third_ovr

    first_list_combine['third_dif'] = third_dif
    first_list_combine['fourth_ovr'] = fourth_ovr

    %snakeviz Update_combine()
    #first_list_combine.head()
```

실행 결과는 아래와 같았다.

몇%를 차지하는 것을 보여주는 것이 도움이 될 것 같다. 성능 개선을 할 때 전체 분석을 해서 실행 시간 비율이 높은 함수를 우선적으로 수정하면 큰 도움이 되겠다.

이렇게 간단하게 python profiler의 하나인 SnakeViz를 직접 설치해보고 프로젝트에 적용도 해보았다. 개발에 있어서 코드를 다 짰다고 끝나는 것이 아니라 이후 유지보수, 성능 개선 등도 정말 중요하다는 것을 수업에서 배웠는데, 그 과정들을 조금이라도 직접 해볼 수 있어서 재밌었다.

<11주차>



➤ 설치 프로그램:

Stacer(<https://oguzhaninan.github.io/Stacer-Web/>)

➤ 프로그램 설명:

Stacer는 오픈소스 기반의 시스템 최적화 및 모니터링 어플리케이션이다. 하나의 utility로 시스템을 여러 관점에서 관리할 수 있게 해준다. 시스템 청소, 서비스/프로세스 관리, 패키지 설치/삭제 등 다양한 기능을 지원한다.

➤ 설치 이유:

이번 주는 이 프로젝트의 마지막 주이다. 그동안 여러 프로그램들을 우분투 위에 설치하였는데, 프로젝트를 마무리한다는 의미로 최적화를 한번 해보기로 하였다. Linux 운영체제에서는 Stacer가 해당 분야에서 꽤나 유명하기 때문에 설치를 진행하게 되었다.

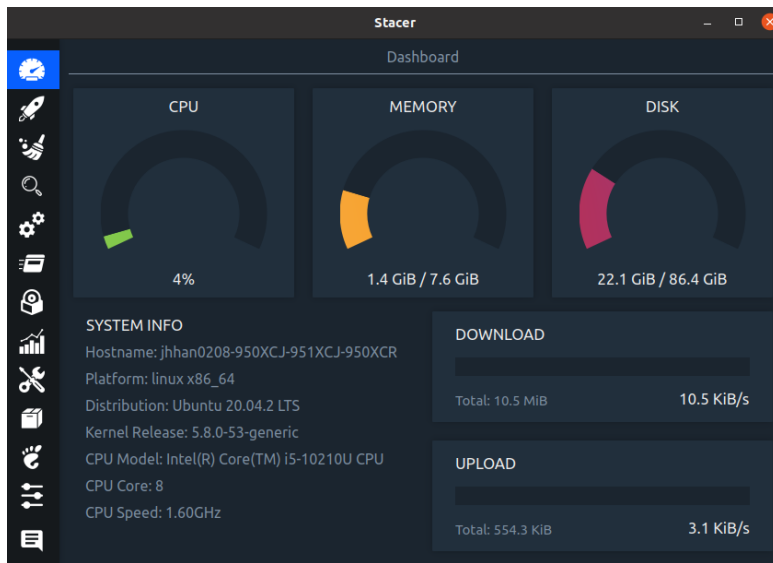
➤ 설치 과정:

Stacer는 Apt 패키지 관리 툴로 쉽게 설치하는 것이 가능하였다. 터미널에 입력을 해서 바로 설치를 진행하였다.

```
(base) jhhan0208@jhhan0208-950XCJ-951XCJ-950XCR:~$ sudo apt install stacer
[sudo] jhhan0208의 암호:
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
```

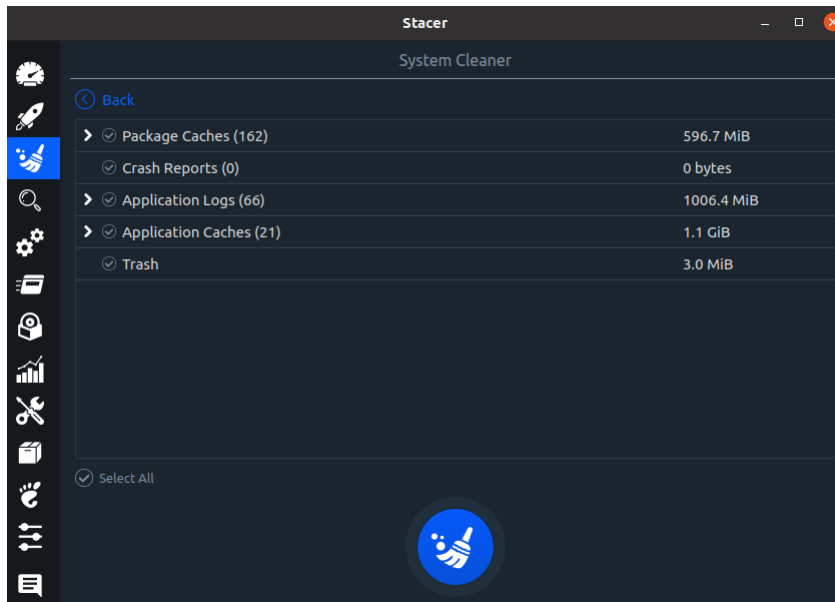
➤ 사용/느낀 점:

우선 설치 후에 Stacer을 실행해보았다.

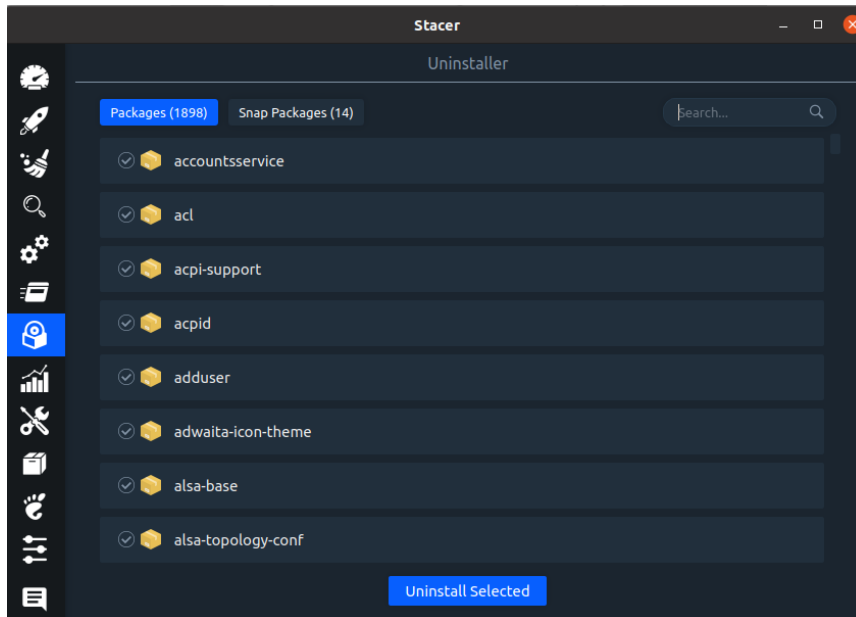


일단 첫 화면은 시스템 최적화 SW가 보편적으로 가지고 있는 UI를 가지고 있음을 알 수 있었다. CPU, 메모리, 디스크의 사용량, 다운로드/업로드 속도 등 여러 시스템 정보들을 표시 해주었다. 동시에 여러 크롬 창을 띄우거나 유튜브 동영상을 틀거나 했을 때 CPU 사용량이 올라가는 점도 재밌었다.

Stacer 공식 홈페이지에서 소개한 주요 기능들을 살펴보기로 하였다. 우선 System Cleaner를 실행해보니 cash 애플리케이션 log, trash 등 여러 종류의 junk file이 각각 얼마나 있는지 보여 주었고, 버튼을 누르면 모두 삭제가 가능했다. 아마 최적화 SW에서 가장 많이 사용하는 기능이 아닐까 싶다.



Uninstaller을 실행하니 그동안 설치했던 패키지, snap 패키지들을 살펴보고, 삭제할 수 있었다. 프로젝트를 진행하면서 이 방식으로 많은 설치를 진행했는데, 그동안 설치한 패키지를 살펴보는 것도 재밌었다. 보면서 불필요한 패키지들은 삭제하였다.



이렇게 Stacer을 통해서 내 linux 운영체제를 직접 최적화해보았다. 그동안 윈도우, 안드로이드 등 어떤 운영체제를 사용해도 최적화는 항상 큰 이슈인 것 같다. 나는 윈도우에서는 고클린이라는 프로그램으로, 안드로이드에서는 삼성의 자체 최적화 기능을 사용했었다. 우분투에서는 최적화를 진행하지 않았는데, 앞으로는 Stacer을 이용하면 되겠다.

<느낀 점>

➤ 1. 배운 점

이 프로젝트를 진행하면서 정말 많은 것을 배웠다. 그 중 중요한 것들만 모아보았다.

- 터미널의 사용

사실 이 프로젝트를 진행하기 전에는 터미널에 대한 지식이 거의 없었다. 지금까지는 에디터 등을 이용해서 운영체제 위에서 개발을 하거나 했지, 그 아래 단계까지는 신경 쓰지 못했다. 그런데 내가 설치한 리눅스의 distro인 우분투에 대해 공부하게 되면서 리눅스에서는 터미널을 정말 많이 사용한다는 것을 알게 되었다.

처음에는 친절한 UI에 비해 모두 손으로 타이핑해야 하는 터미널이 복잡하고 어렵기만 했다. 그런데 여러 방식으로 프로그램들을 설치해보면서 터미널의 사용에 익숙해졌고, 어떤 면에서는 UI방식으로 설치하는 것보다 더 편리하다고 느끼기도 하였다.

- 다양한 운영체제

이전에 나는 무의식 중에 “컴퓨터 = 윈도우”라는 생각을 계속 해왔던 것 같다. 내가 쓰고 있는 삼성 컴퓨터, 노트북이 default로 윈도우 기반으로 돌아가기 때문이기도 하겠지만, 그냥 다른 운영체제들은 배제했던 것 같다. 그리고 윈도우 이외의 운영체제에 대해서는 ‘굳이’라는 생각으로 관심을 별로 가지지 않았다. 그런데 우분투를 한학기 사용해보면서 윈도우가 절대적이라는 나의 생각은 꽤나 약해졌다. 물론 윈도우에서 이전에 사용하던 프로그램들 중에 리눅스에서는 돌아가지 않는 것들도 있었지만, 대부분의 경우에는 같은 역할을, 심지어 더 잘하는 대체 sw가 존재하였다.

또한 우분투라는 운영체제 자체에 대해서도 애정을 좀 더 가지게 된 것 같다. 아이콘들이 아기자기한 느낌이었고 무엇보다 전반적인 UI가 깔끔해서 좋았다. 그리고 프로그램 설치, 삭제, 업데이트 등도 터미널을 통해서 더 간단하게 할 수 있었다. 이후에도 리눅스 mint 등 리눅스의 다른 distro들도 사용해보고 싶다.

➤ 2. 소감

우선 11주, 거의 한학기 동안 진행되었던 “리눅스로 한학기 살기” 프로젝트를 무사히 마치게 되어서 정말 뿌듯하다. 프로젝트 초기에는 정말 이걸 할 수 있을지에 대한 생각부터 해서 정말 막막했다. 그러나 매주 시간을 계속 투자하니 프로그램을 찾아 설치하고 사용하기까지의 일련의 과정에 점점 익숙해졌고, 자연스러워진 것 같다.

우분투를 설치하고 10개정도의 프로그램을 설치하니 그냥 사용하기에 무리가 없는 환경을 제법 갖춘 느낌이다. 내가 생각했던 3가지 concept에 따라서 설치하니 적어도 그 부분에 대해서는 부족함이 없었다. 컴퓨터는 놔두고 노트북에서 윈도우와의 듀얼 부팅 방식으로 이 프로젝트를 진행했는데, 노트북에서는 윈도우를 삭제하고 우분투 단일체제로 가는 것도 나쁘지 않을 것 같다. 초기에는 윈도우의 다양한 기능을 다 옮겨올 수 있을지에 대한 의심이 있었지만, 상당 부분에서는 대체가 가능했고, 우분투만의 기능 중에서도 매력적인 것이 많았다. 특히 개발을 함에 있어서는 윈도우보다 좋았다.

하나의 과제로써 진행한 프로젝트였지만 정말 재밌었다. 매주 새로운 프로그램으로 컴퓨터가 업그레이드되는 느낌, 그리고 매주 내 리눅스 지식이 조금이지만 업그레이드되는 느낌이 모두 좋았다. 이렇게 오랜 시간 투자한 프로젝트는 처음이지만, 투자한 시간 만큼 의미가 있었다고 생각한다. 종강 이후에도 나는 계속해서 리눅스를 main OS로 사용할 계획이다.