

CLI 기초교육

서버에서 길잃지 않기

할 일 목록

1. Linux가 무엇인지 읽어보기
2. Terminal과 shell의 차이점 알아보기
3. 명령어의 본질
4. sh 스크립트
5. CLI 환경에서의 작업
6. Python과 가상환경에 대해서
7. 그 외
 - A. 설명하지 않은 기초 명령어
 - B. vi 초간단 사용법
 - C. vi가 비정상 종료 되었을 때

Linux란 무엇인가

리눅스(영어: **Linux**)는 리누스 토르발스가 커뮤니티 주체로 개발한 컴퓨터 운영 체제이다. 혹은 커널을 뜻하기도 한다. 리눅스는 자유 소프트웨어와 오픈 소스 개발의 가장 유명한 표본으로 들 수 있다. 리눅스는 다중 사용자, 다중작업 (멀티태스킹), 다중 스레드를 지원하는 네트워크 운영 체제(**NOS**)이다.

엄밀하게 따지면 이 ‘리눅스’라는 용어는 리눅스 커널만을 뜻하지만, 리눅스 커널과 **GNU** 프로젝트의 라이브러리와 도구들이 포함된, 전체 운영 체제 (**GNU/리눅스**라고도 알려진)를 나타내는 말로 흔히 쓰인다. 리눅스 배포판은 핵심 시스템 외에 대다수 소프트웨어를 포함한다. 현재 200여 종류가 넘는 배포판이 존재한다.

출처: Wikipedia

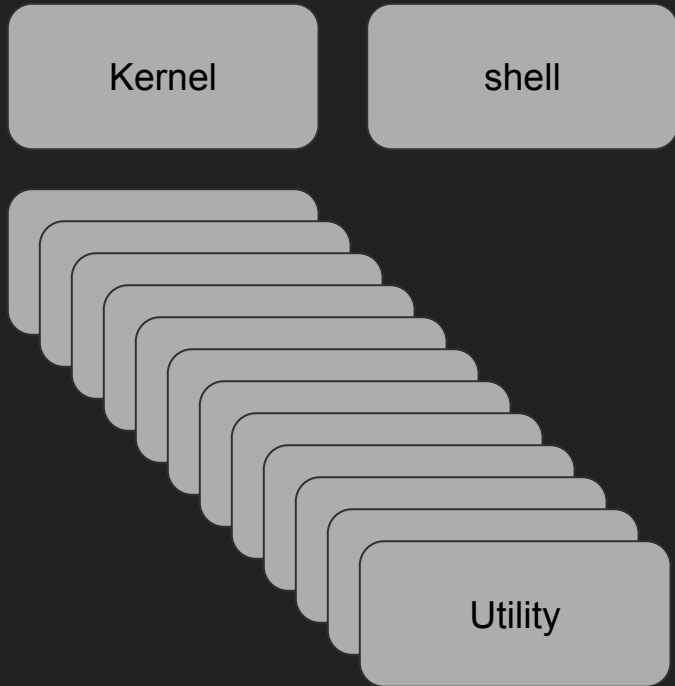
Linux란 무엇인가

커널의 일종인 리눅스 커널, 또는 리눅스 커널을 사용하는 운영 체제를 가리키는 말이기도 하다. **GNU** 쪽 사람들은 리눅스는 커널일 뿐이고, 이 커널을 가져다가 **GNU** 프로그램들을 올려 만든 운영 체제는 **GNU/Linux**라고 이야기하며 이런 명칭에 민감하게 반응하는 경우도 있다. 소스 코드가 공개되어 있는 대표적인 오픈 소스 소프트웨어다. 컴퓨터 역사상 가장 많은 사람이 들어간 오픈 소스 프로젝트이며, 모바일 운영 체제로 유명한 안드로이드가 이것을 기반으로 한다.

Linux라는 이름은 **Linus' *nix**(리누스의 유닉스)라는 뜻으로 지어졌다. ***nix**는 **Unix** 계열 운영체제라는 뜻이다. 나중에 **Linux Is Not Unix**라는 재귀적 용어의 줄임말이라는 의미를 새로 만들어냈다.

출처: 나무위키

Linux란 무엇인가

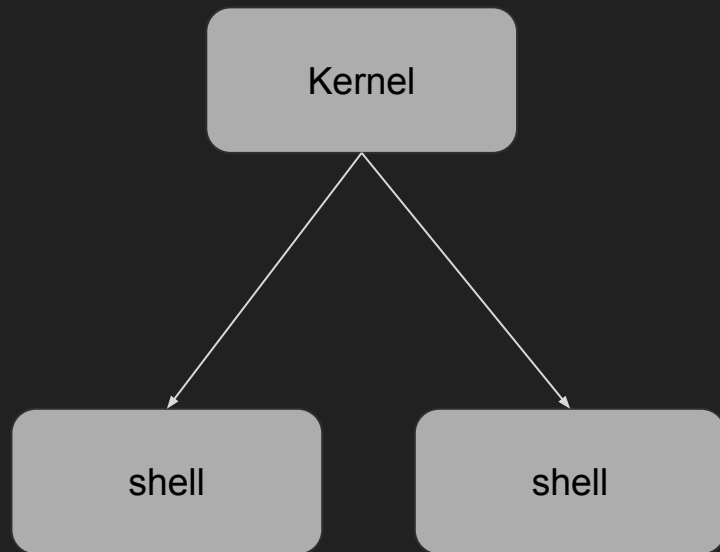
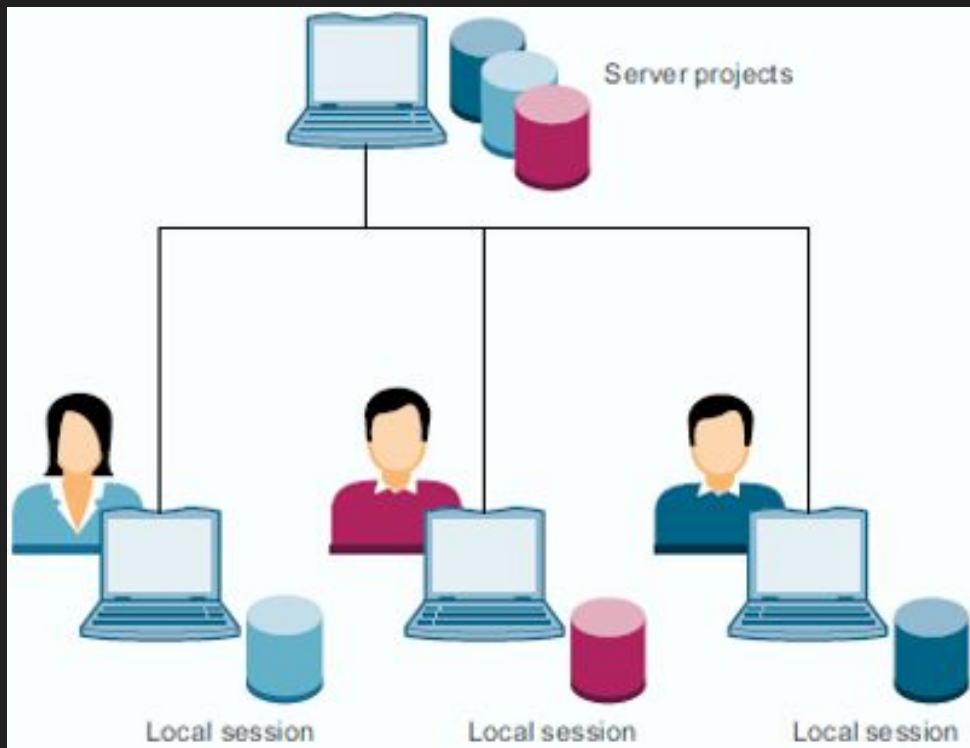


Kernel: Linux의 본질, 하드웨어 메모리를 비롯한 것을 관리

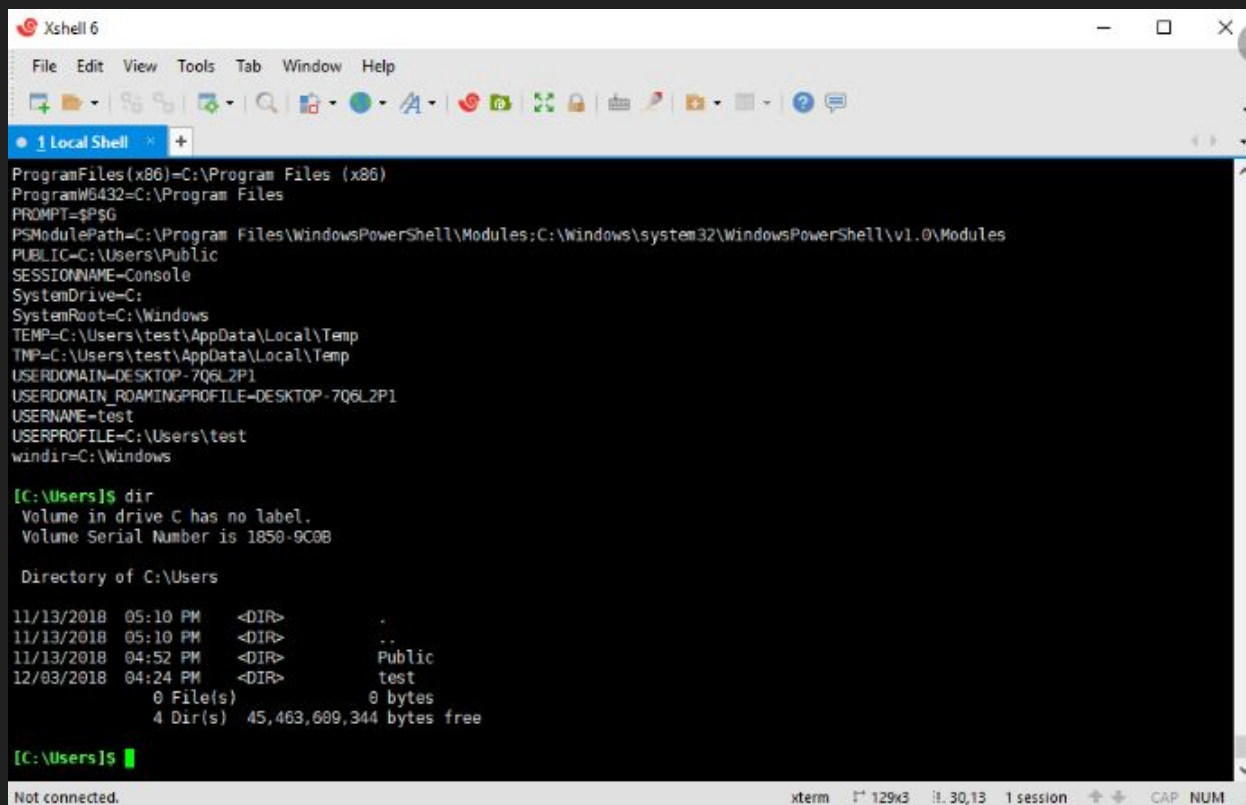
shell: OS를 사용자가 사용하기 위한 UI

Utility: 그 외에 사용자가 사용하는 명령어 내지 프로그램 들

Linux란 무엇인가



Terminal?



Xshell 6

File Edit View Tools Tab Window Help

1 Local Shell

```
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$PSG
PSModulePath=C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
PUBLIC=C:\Users\Public
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\test\AppData\Local\Temp
TMP=C:\Users\test\AppData\Local\Temp
USERDOMAIN=DESKTOP-7Q6L2P1
USERDOMAIN_ROAMINGPROFILE=DESKTOP-7Q6L2P1
USERNAME=test
USERPROFILE=C:\Users\test
windir=C:\Windows
```

[C:\Users] dir

Volume in drive C has no label.
Volume Serial Number is 1850-9C0B

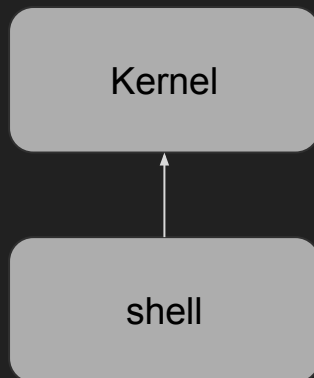
Directory of C:\Users

11/13/2018	05:10 PM	<DIR>	.
11/13/2018	05:10 PM	<DIR>	..
11/13/2018	04:52 PM	<DIR>	Public
12/03/2018	04:24 PM	<DIR>	test
		0 File(s)	0 bytes
		4 Dir(s)	45,463,609,344 bytes free

[C:\Users] \$

Not connected. xterm 129x3 11.30,13 1 session CAP NUM

Terminal!



A screenshot of an Xshell 6 terminal window. The window title is "Xshell 6". The menu bar includes File, Edit, View, Tools, Tab, Window, and Help. The toolbar contains various icons for file operations and terminal control. The terminal output shows system environment variables and a directory listing:

```
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$P$G
PSModulePath=C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
PUBLIC=C:\Users\Public
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\Windows
TEMP=C:\Users\test\AppData\Local\Temp
TMP=C:\Users\test\AppData\Local\Temp
USERDOMAIN=DESKTOP-7Q6L2P1
USERDOMAIN_ROAMINGPROFILE=DESKTOP-7Q6L2P1
USERNAME=root
USERPROFILE=C:\Users\test
windir=C:\Windows

[C:\Users]$ dir
Volume in drive C has no label.
Volume Serial Number is 1850-9C0B

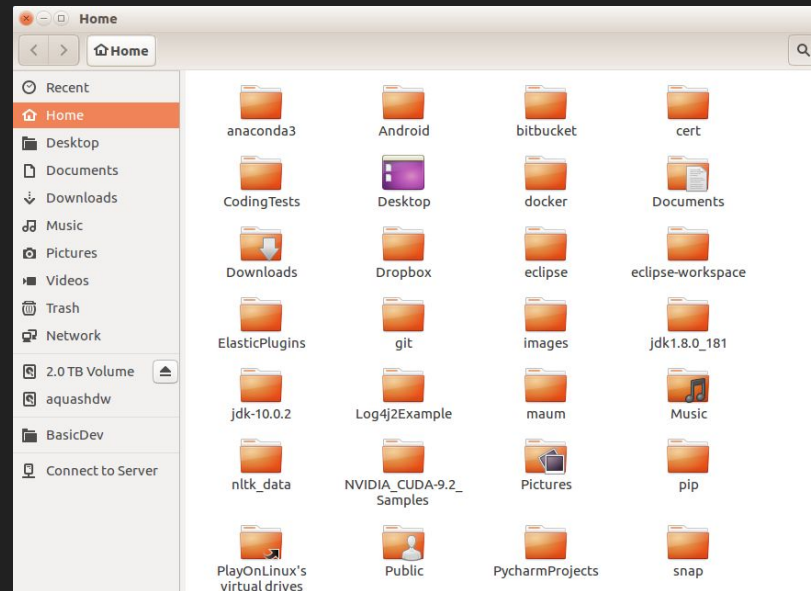
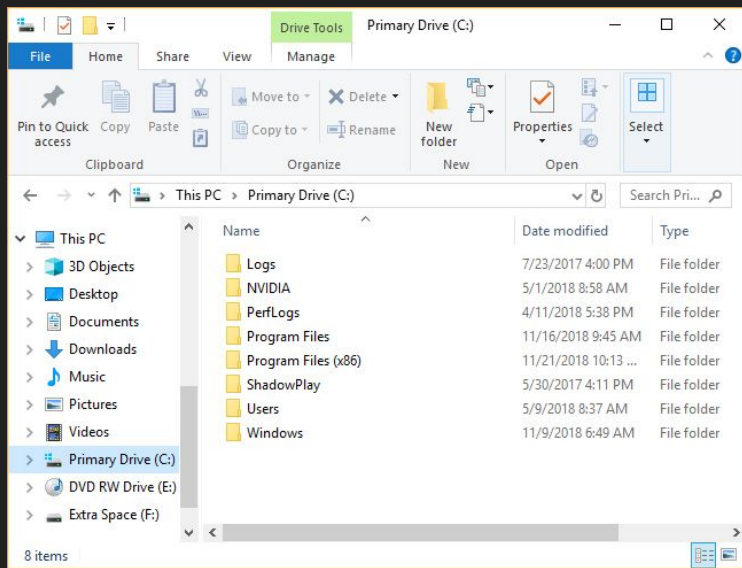
Directory of C:\Users

11/13/2018  05:10 PM  <DIR>      .
11/13/2018  05:10 PM  <DIR>      ..
11/13/2018  04:52 PM  <DIR>      Public
12/03/2018  04:24 PM  <DIR>      test
               0 File(s)                0 bytes
               4 Dir(s)  45,463,669,344 bytes free

[C:\Users]$
```

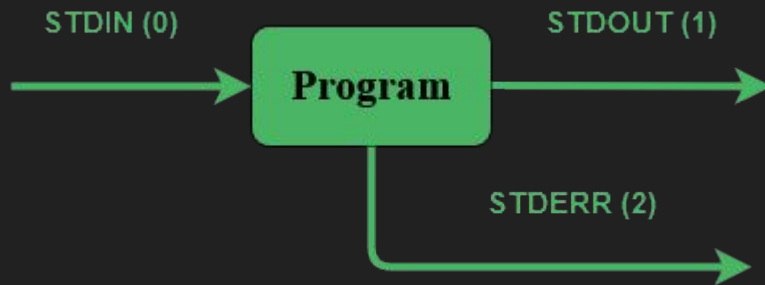
The status bar at the bottom indicates "Not connected." and "xterm 129x3 11.30.13 1 session GAP NUM".

Explorer & Nautilus



명령어의 본질

```
→ ~ ps -ef █  
  
→ ~ ffmpeg -i test.wav -ss 10 -t 5 asdf.wav █  
  
→ tts-stream-test java -jar target/tts-stream-test.jar  
→ ~ man man █  
  
→ ~ pip install pydub █
```



명령어의 본질

헛갈리지 말아야 할 것

CLI에서의 입출력과
프로그램의 입출력은 다르다

명령어의 본질



문자열 != STDIN
음성 != STDOUT | STDERR

명령어를 연속으로 사용할 때...

- `expr1 & expr2` : `expr1` 이후에 `expr2`
- `expr1 && expr2` : `expr1` 성공시(exit code 0) `expr2`
- `expr1 > object_name` : `expr1`의 `stdout` / `stderr`를 `object_name`으로 redirect
- `expr1 >> object_name` : `expr1`의 `stdout` / `stderr`를 `object_name`으로 redirect & append
- `expr1 | expr2`: `expr1`의 `stdout` / `stderr`를 `expr2`의 **stdin**으로 보낸다.

sh 스크립트

shell: OS를 사용자가 사용하기 위한

UI
shell script: 셸이나 명령 줄 인터프리터에서 돌아가도록 작성되었거나 한 운영 체제를 위해 쓰인 스크립트

간단히 말하면 명령어
모음집

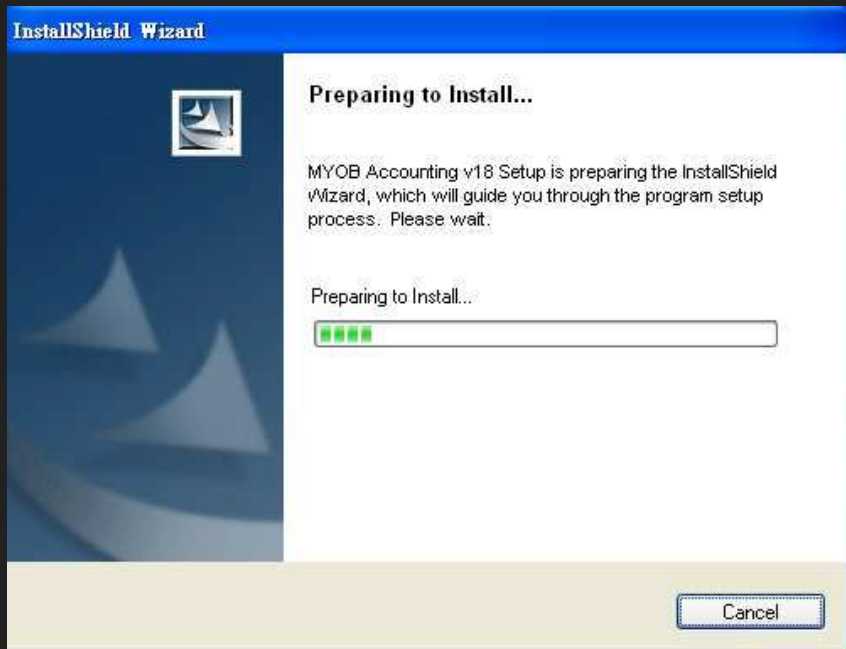
```
export MEM_ARGS="-verbosegc -Xms512m -Xmx4096m"
export P_NM="tts_rest_${TTS_PROFILE}_${hostname}"

echo "profile ${TTS_PROFILE}"

count=`ps -ef | grep ${P_NM} |grep -v grep |awk '{print $2}'|wc -l`

if [ $count -gt 0 ]; then
    echo "${P_NM} Process is already running ...";
else
    ${JAVA_HOME}/bin/java -jar -DNODE_NAME=${P_NM} \
    -Dlog4j.configurationFile=file://${MAUM_ROOT}/etc/brain-tts/spring-boot/tts-rest/
    /log4j2.xml \
    -Dspring.config.location=file:${MAUM_ROOT}/etc/brain-tts/spring-boot/tts-rest/ap
    plication-${TTS_PROFILE}.properties \
    -Dspring.profiles.active=${TTS_PROFILE} \
    ${MAUM_ROOT}/lib/tts-rest.war ${MEM_ARGS} > /dev/null 2>&1 &
fi
```

sh 스크립트



```
# -----  
# Ensure IDE_HOME points to the directory where the IDE is installed.  
# -----  
SCRIPT_LOCATION="$0"  
if [ -x "$READLINK" ]; then  
    while [ -L "$SCRIPT_LOCATION" ]; do  
        SCRIPT_LOCATION=$( "$READLINK" -e "$SCRIPT_LOCATION" )  
    done  
fi  
  
cd "$( "$DIRNAME" "$SCRIPT_LOCATION" )" || exit 2  
IDE_BIN_HOME=$(pwd)  
IDE_HOME=$( "$DIRNAME" "$IDE_BIN_HOME" )  
cd "${OLDPWD}" || exit 2  
# -----  
# Locate a JDK installation directory command -v will be used to run the IDE  
# Try (in order): RIDER_JDK, rider.jdk, ./jbr, ./jre64, JDK_HOME, JAVA_HOME  
# -----  
# shellcheck disable=SC2154  
if [ -n "$RIDER_JDK" ] && [ -x "$RIDER_JDK/bin/java" ]; then
```

환경 변수

환경 변수(環境變數, 영어: **environment variable**)는 프로세스가 컴퓨터에서 동작하는 방식에 영향을 미치는, 동적인 값들의 모임이다.

출처: Wikipedia

```
→ ~ echo $PATH
/home/aquashdw/Downloads/google-cloud-sdk/bin:/home/aquashdw/Documents/arrayfire
:/home/aquashdw/intel/mkl:/usr/local/cuda-8.0/bin:/opt/gradle/gradle-4.10.1/bin:
/home/aquashdw/Documents/flutter/bin:/home/aquashdw/git:/bin:/home/aquashdw/Docu
ments/Postman:/home/aquashdw/Documents/kibana-6.4.1-linux-x86_64//bin:/home/aqua
shdw/Documents/elasticsearch-6.4.0/bin:/home/aquashdw/anaconda3/bin:/home/aquash
dw/bin:/home/aquashdw/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/aquashdw/.dotnet/tools
:/home/aquashdw/Documents/flutter/.pub-cache/bin
```


환경변수

동작하는 방식에 영향을 → 원하는 대로 작동하기 위해 알아야

명확한 값들의 → 변할 수 있다

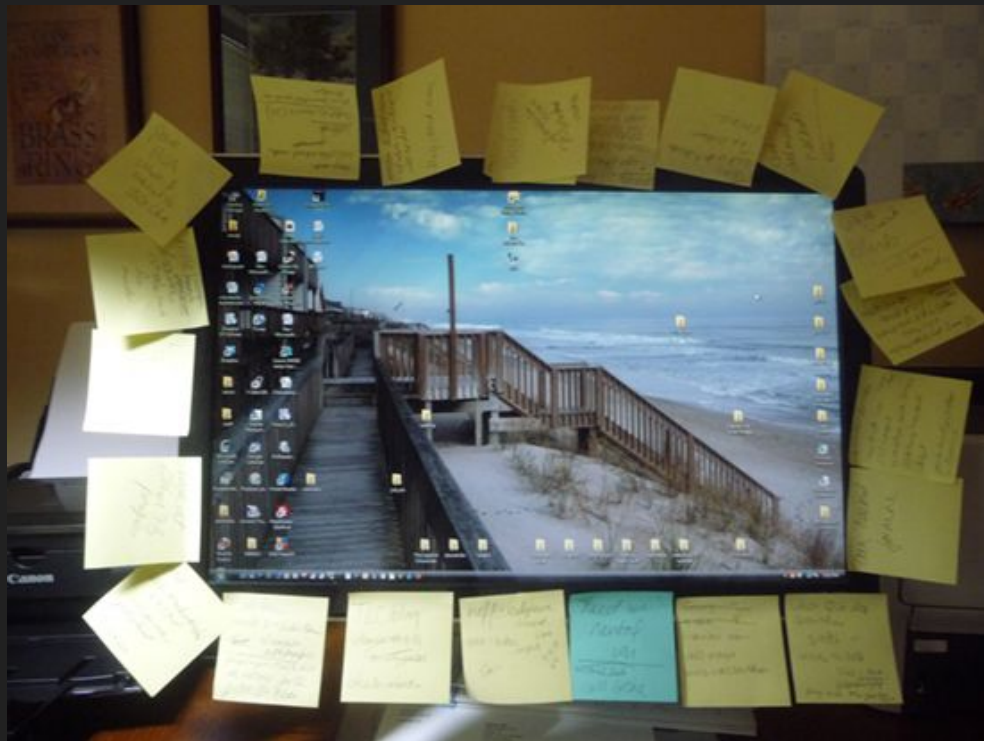
모임

```
→ ~ echo $PATH
/home/aquashdw/Downloads/google-cloud-sdk/bin:/home/aquashdw/Documents/arrayfire
:/home/aquashdw/intel/mkl:/usr/local/cuda-8.0/bin:/opt/gradle/gradle-4.10.1/bin:
/home/aquashdw/Documents/flutter/bin:/home/aquashdw/git:/bin:/home/aquashdw/Docu
ments/Postman:/home/aquashdw/Documents/kibana-6.4.1-linux-x86_64/bin:/home/aqua
shdw/Documents/elasticsearch-6.4.0/bin:/home/aquashdw/anaconda3/bin:/home/aquash
dw/bin:/home/aquashdw/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/aquashdw/.dotnet/tools
:/home/aquashdw/Documents/flutter/.pub-cache/bin
```

```
→ ~ export PATH="/usr/bin:/usr/local/bin"
→ ~ echo $PATH
/usr/bin:/usr/local/bin
```

환경변수

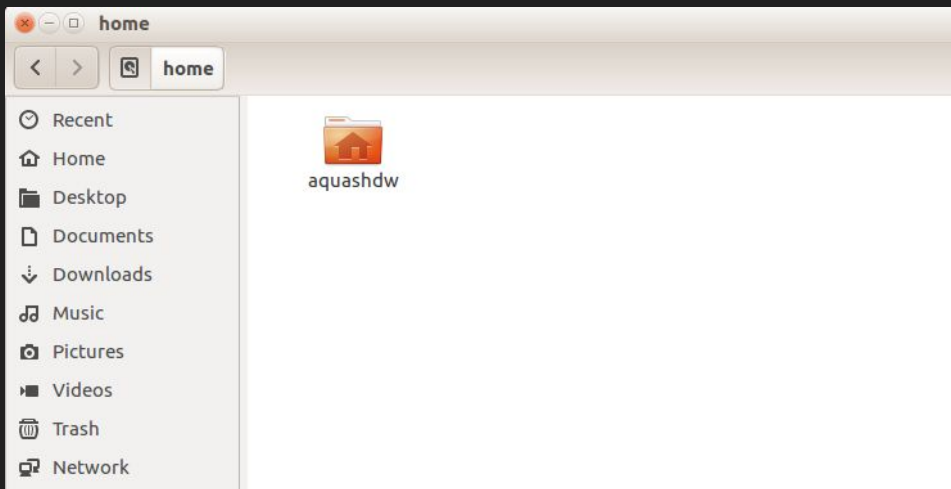
필요할 때 확인하는
메모지 같은 것



환경변수는 어디서?

home에 대하여

```
→ ~ pwd  
/home/aquashdw
```



환경변수는 어디서?

사용자 환경
변수

```
→ ~ pwd
/home/aquashdw
-rw-r--r--  1 aquashdw aquashdw      6119  7월  17  2019  .bashrc
-rw-r--r--  1 aquashdw aquashdw      3855  9월  17  2018  .bashrc-anaconda3.bak
-rw-r--r--  1 aquashdw aquashdw      8276  3월  2  17:19  .zshrc
-rw-rw-r--  1 aquashdw aquashdw      7698  1월  8  19:14  .zshrc.backup
-rw-r--r--  1 aquashdw aquashdw      1295 12월 14  2018  .zshrc.pre-oh-my-zsh

# flutter path
export PATH="/home/aquashdw/Documents/flutter/bin:$PATH"
export PATH="$PATH":"$HOME/Documents/flutter/.pub-cache/bin"

# gradle path
export PATH="/opt/gradle/gradle-4.10.1/bin:$PATH"

# CUDA path
export PATH="/usr/local/cuda-8.0/bin${PATH:+:${PATH}}"
export LD_LIBRARY_PATH="/usr/local/cuda-8.0/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}"
```

환경변수는 어디서?

시스템(컴퓨터) 환경 변수

```
→ ~ vi /etc/profile
→ ~ █
if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

Python!

Python

파이썬(영어: Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어

Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

출처: Wikipedia

pip는 Python 용 패키지 설치 프로그램입니다. pip를 사용하여 Python 패키지 색인 및 기타 색인에서 패키지를 설치할 수 있습니다.

출처: <https://pypi.org/project/pip/>

pip!

```
→ ~ pip install pydub
Collecting pydub
  Using cached https://files.pythonhosted.org/packages/79/db/eaf620b73a1eec3c8c6f8f5b0b236a50f9da88ad57802154b7ba7664d0b8/pydub-0.23.1-py2.py3-none-any.whl
Installing collected packages: pydub
Successfully installed pydub-0.23.1
```

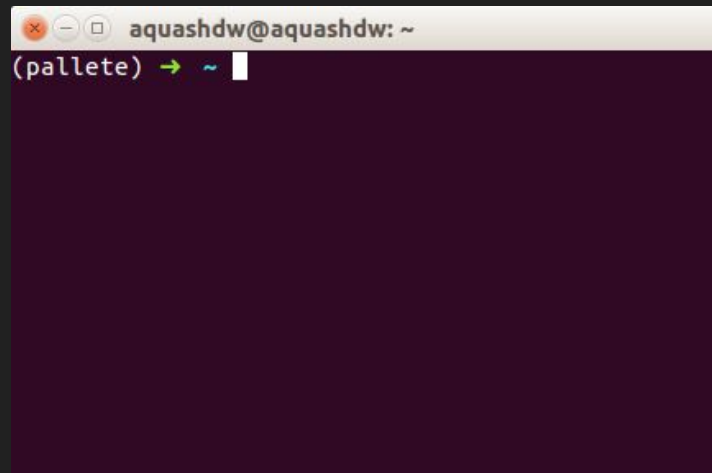
```
drwxrwxr-x  2 aquashdw aquashdw    4096  9월 17  2018 xlwt-1.3.0-py3.6.egg-info
/
drwxrwxr-x  2 aquashdw aquashdw    4096  1월 22  2019 xonsh/
drwxrwxr-x  3 aquashdw aquashdw    4096  9월 17  2018 yaml/
-rwxrwxr-x  2 aquashdw aquashdw 233040  9월 18  2017 _yaml.cpython-36m-x86_64-
linux-gnu.so*
drwxrwxr-x  3 aquashdw aquashdw    4096  9월 17  2018 zict/
drwxrwxr-x  2 aquashdw aquashdw    4096  9월 17  2018 zict-0.1.3-py3.6.egg-info
/
drwxrwxr-x 14 aquashdw aquashdw    4096  9월 17  2018 zmq/
```

```
→ site-packages pwd
/home/aquashdw/anaconda3/lib/python3.6/site-packages
```

```
→ site-packages
```

```
→ site-packages ll | grep pydub
drwxrwxr-x  3 aquashdw aquashdw    4096  4월 27 14:50 pydub/
drwxrwxr-x  2 aquashdw aquashdw    4096  4월 27 14:50 pydub-0.23.1.dist-info/
```

Python 가상환경



Python 가상환경

```
→ ~ source .virtualenvs/pallete/bin/activate  
(pallete) → ~ █
```

source: 인자로 받은 shell script 파일을
현재의 세션에서 실행한다.

그 외

grep : 주어진 파일에서 주어진 패턴을 검색해서 그 줄을 출력하는 명령어. 입력 파일이 주어지지 않을 경우 **stdin**을 검색한다.

ps : 현재의 프로세스들의 **snapshot**을 출력하는 명령어. 주로 **-ef** (e: all processes, f: full-format) 옵션을 함께 사용한다.

```
ps -ef | grep java
```

그 외

grep : 주어진 파일에서 주어진 패턴을 검색해서 그 줄을 출력하는 명령어. 입력 파일이 주어지지 않을 경우 **stdin**을 검색한다.

netstat : Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships

```
netstat -nlpt | grep java
```

그 외

tail : 주어진 파일의 마지막 10줄을 출력한다. (f: 파일에 **append**가 일어날 때 추가로 출력한다.)

```
ubuntu@ip-172-31-25-47:~/maum/logs/rest$ tail -f brain.api.server.log
INFO | 2020-04-27 15:09:55 | [TTS] {Controller} [TTS Stream] : {apiId=neocomix,
apiKey=328239e0cac840929309d5de7faa09d0, voiceName=neocomix_ktj, text=연기됐던
한미연합 공중훈련, 20일부터 전격 재개.}
WARN | 2020-04-27 15:09:55 | [COMMON] {Service} user agent not defined
WARN | 2020-04-27 15:09:55 | [COMMON] {Service} Argument 'userAgentString' must
not be null.
INFO | 2020-04-27 15:09:55 | [TTS] {Service} TTS Done User : neocomix voiceName
: neocomix_ktj
INFO | 2020-04-27 15:09:56 | [TTS] {Controller} [TTS Stream] : {apiId=neocomix,
apiKey=328239e0cac840929309d5de7faa09d0, voiceName=neocomix_ktj, text=다음주부
터 공적마스크 1인 3매로. 한국전 참전용사에도 지원.}
WARN | 2020-04-27 15:09:56 | [COMMON] {Service} user agent not defined
WARN | 2020-04-27 15:09:56 | [COMMON] {Service} Argument 'userAgentString' must
not be null.
INFO | 2020-04-27 15:09:56 | [TTS] {Service} TTS Done User : neocomix voiceName
: neocomix_ktj
INFO | 2020-04-27 15:10:00 | [TTS] {Controller} [TTS Stream] : {text=안녕하세요
., voiceName=baseline_kor, apiId=maum-ai-web-demo, apiKey=eee2ed4e7e084c32adc2ad
93d14bb22a}
INFO | 2020-04-27 15:10:00 | [TTS] {Service} TTS Done User : maum-ai-web-demo v
oiceName : baseline_kor
```

그 외

top

```
top - 15:02:04 up 5:52, 2 users, load average: 0.37, 0.47, 0.53
Tasks: 446 total, 1 running, 329 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.1 us, 3.2 sy, 0.0 ni, 93.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 32954976 total, 17000644 free, 8285216 used, 7669116 buff/cache
KiB Swap: 998396 total, 998396 free, 0 used, 23043764 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1293	root	20	0	502304	162160	55620	S	36.8	0.5	17:29.88	Xorg
2154	aquashdw	20	0	1885948	542672	104412	S	13.6	1.6	48:46.54	compiz
7531	aquashdw	20	0	4999708	198736	88980	S	12.3	0.6	3:42.26	chrome
16331	aquashdw	20	0	4946612	196492	86004	S	9.3	0.6	33:47.83	chrome
7077	aquashdw	20	0	2876780	172656	91116	S	4.6	0.5	37:22.35	clementine
1814	root	20	0	920456	61640	50840	S	3.6	0.2	0:00.11	kubelet
1801	aquashdw	20	0	435036	22520	19044	S	3.0	0.1	0:00.15	gnome-scre
2093	aquashdw	9	-11	577424	15128	11424	S	1.7	0.0	6:38.04	pulseaudio
7407	aquashdw	20	0	994308	443944	140500	S	1.7	1.3	13:21.39	chrome
2857	root	20	0	1313984	76284	30284	S	1.3	0.2	5:36.15	dockerd
1842	mssql	20	0	18.552g	1.034g	58848	S	1.0	3.3	3:56.13	sqlservr
5423	aquashdw	20	0	23.477g	733800	254844	S	1.0	2.2	7:28.64	slack
7367	aquashdw	20	0	1599432	367308	153276	S	1.0	1.1	11:03.62	chrome
26737	aquashdw	20	0	4933152	136344	87480	S	1.0	0.4	0:31.06	chrome
1597	aquashdw	20	0	35256	276	12	S	0.7	0.0	0:23.67	upstart-ud
13432	aquashdw	20	0	10.150g	0.996g	84468	S	0.7	3.2	4:47.83	java
1239	root	20	0	830536	35260	26232	S	0.3	0.1	0:11.03	libvirt

htop

```
htop
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	Command
1845	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.22	/opt/mssql/bin/sq
1846	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.00	/opt/mssql/bin/sq
1847	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.07	/opt/mssql/bin/sq
1848	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:02.51	/opt/mssql/bin/sq
1851	mssql	20	0	18.6G	1059M	58848	S	1.3	3.3	2:17.31	/opt/mssql/bin/sq
1852	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.34	/opt/mssql/bin/sq
1853	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.02	/opt/mssql/bin/sq
1854	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.00	/opt/mssql/bin/sq
1856	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.02	/opt/mssql/bin/sq
1858	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.01	/opt/mssql/bin/sq
1859	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.00	/opt/mssql/bin/sq
1860	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.00	/opt/mssql/bin/sq
1861	mssql	20	0	18.6G	1059M	58848	S	0.0	3.3	0:00.02	/opt/mssql/bin/sq

F1 help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort By F7 Nice F8 Nice F9 Kill F10 Quit

그 외

watch : 주기적으로 프로그램을 실행하여 전체 화면으로 결과를 보여준다.

nvidia-smi

```
Mon Apr 27 15:04:36 2020
+-----+
| NVIDIA-SMI 410.48                  Driver Version: 410.48                  |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0   Tesla V100-PCIE...    Off   | 00000000:18:00:0 Off |             0      |
| N/A   41C    P0     37W / 250W | 8977MiB / 32480MiB |      0%    Default  |
+-----+-----+
|  1   Tesla V100-PCIE...    Off   | 00000000:AF:00:0 Off |             0      |
| N/A   39C    P0     37W / 250W | 1184MiB / 32480MiB |      0%    Default  |
+-----+-----+

+-----+-----+
| Processes:                         GPU Memory |
|   GPU       PID    Type    Process name      Usage   |
+-----+-----+
|    0      106567    C     python             6299MiB |
|    0      209652    C     python             2667MiB |
|    1      209651    C     python             1173MiB |
+-----+-----+
```

watch -n1 nvidia-smi

설명하지 않은 기초 명령어

`pwd`: present working directory

`man`: an interface to the on-line reference manuals

`cd`: change directory

`ls`: list directory contents

`touch`: change file timestamps (파일을 생성할 때 사용)

`vi(vim)`: text editor

설명하지 않은 기초 명령어

tail: print the last 10 lines of each FILE to stdout

head: print the first 10 lines of each FILE to stdout

cat: concatenate FILE to stdout (file의 내용을 stdout으로)

wc: print newline, word, and byte counts for each FILE (file의 개행, 줄, 바이트를 출력, FILE이 안주어질 경우 stdin)

초간단 vi 사용법

실제로 사용되는건 순수 vi가 아닌 업데이트 버전 vim이라는 점을 염두에 두세요.
서버에서 직접 코딩할일은 없으실태니 가장 간단하게 설명 드리겠습니다.

실행: vi (파일명)

명령 입력 상태에서,

i : (누르는 즉시) 수정 시작, w : 저장, q : 종료, q! : 저장하지 않고 종료, wq :
저장하고 종료, dd : (현재 커서 위치에)(누르는 즉시) 줄 삭제

수정 상태에서,

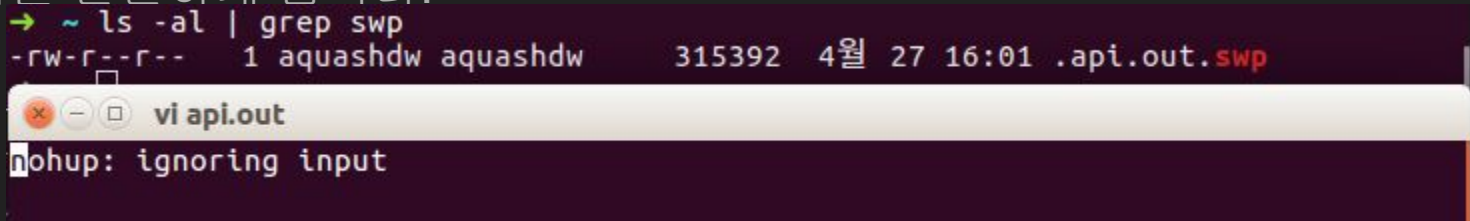
평범한 에디터 처럼 쓰다가 esc를 누르면 다시 명령 입력 상태로

vi가 비정상 종료 되었을 때

vi는 즉시 파일을 변형시키지 않고, **cache-and-commit** (이라고 제가 부르는) 방식으로 파일을 수정하는 프로그램입니다. 이를 위해 **.swp** 이라는 파일을 생성하고, 이곳에 수정을 가한뒤 저장(**write**) 명령을 통해 본래의 파일에 수정사항을 적용하게 됩니다.

vi가 비정상 종료될 경우 이 **.swp** 파일이 제대로 제거되지 않게 되며, 이 파일이 존재할 경우 다른 세션의 vi 는 누군가가 이 파일을 사용하고 있다고 생각하게 되며, 메시지를 반환하게 됩니다.

```
→ ~ ls -al | grep swp
-rw-r--r--  1 aquashdw aquashdw   315392  4월 27 16:01 .api.out.swp
```



The image shows a terminal window with a dark purple background. The top part displays the command `ls -al | grep swp` and its output, which lists a file named `.api.out.swp` with permissions `-rw-r--r--`, owned by `aquashdw`, and a size of `315392`. Below the terminal output, there is a window titled `vi api.out`. This window has a light yellow title bar and a dark purple body. Inside the window, the text `nohup: ignoring input` is visible.

vi가 비정상 종료 되었을 때

자신이 수정을 가하다 vi가 비정상 종료되었을 때, `ls -al` 을 통해 (`a` 옵션은 `'`으로 시작하는 숨겨진 파일도 나열합니다) `swap` 파일의 존재를 확인하고, 해당 파일을 사용하는 vi 프로세스가 있는지 확인하고, (자신이 실행한 프로세스인지 확인을 해보신 후) 해당 프로세스를 멈추고 파일을 지우면 정상적으로 다시 작동하게 됩니다.

```
→ ~ lsof .api.out.swp
lsof: WARNING: can't stat() nsfs file system /run/docker/netns/ingress_sbox
Output information may be incomplete.
lsof: WARNING: can't stat() nsfs file system /run/docker/netns/1-zdbi3gghoo
Output information may be incomplete.
COMMAND PID      USER      FD  TYPE DEVICE SIZE/OFF      NODE NAME
vi        624 aquashdw   4u   REG   8,1    315392 8666330 .api.out.swp
→ ~ █
```