

리눅스, CLI 환경

세종대학교 이은상

CLI 사용시 유의사항

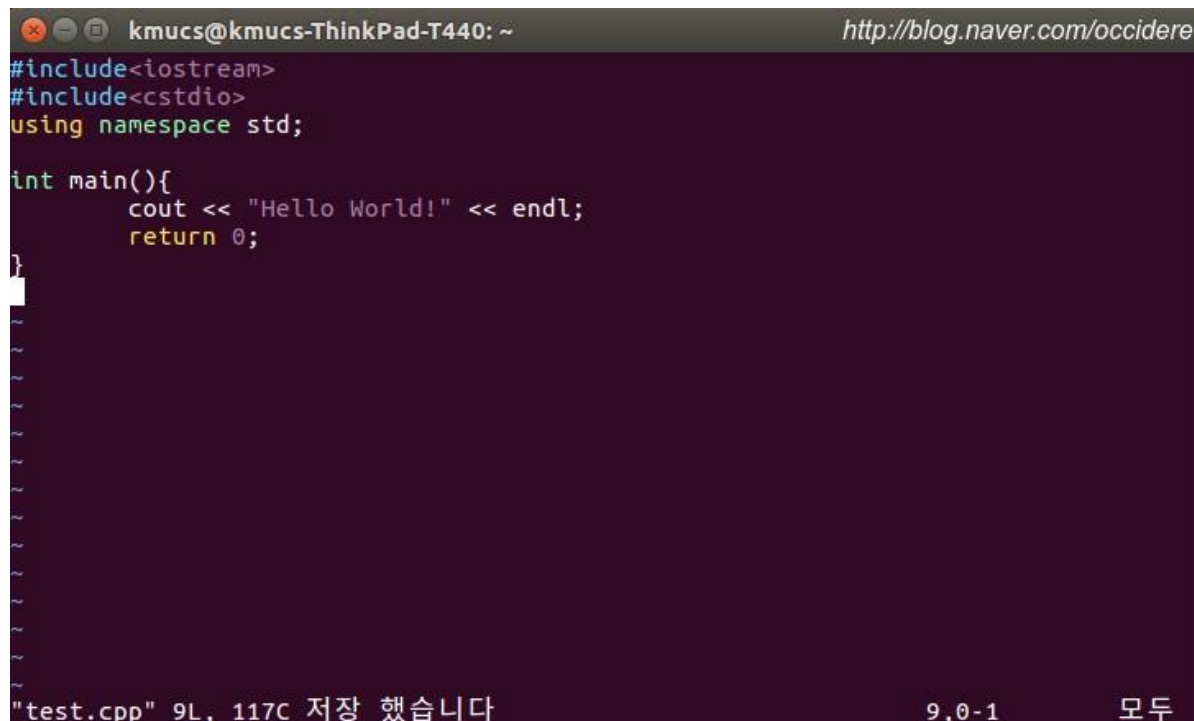
- 오류메시지를 꼼꼼히 읽어보는 습관 필요

```
for@Eunsang MINGW64 ~/Programming
$ git it is
git: 'it' is not a git command. See 'git --help'.

The most similar command is
    init
```

VI 편집기

- Vi는 리눅스에서 많이 사용되는 텍스트 편집기 중 하나
- visual editor의 약자
- Vi는 유닉스 시스템에서 처음 개발되었으며 이후 많은 리눅스 배포판에서 기본적으로 제공됨
- 커맨드라인에서 작동



```
kmucs@kmucs-ThinkPad-T440: ~ http://blog.naver.com/occidere
#include<iostream>
#include<cstdio>
using namespace std;

int main(){
    cout << "Hello World!" << endl;
    return 0;
}

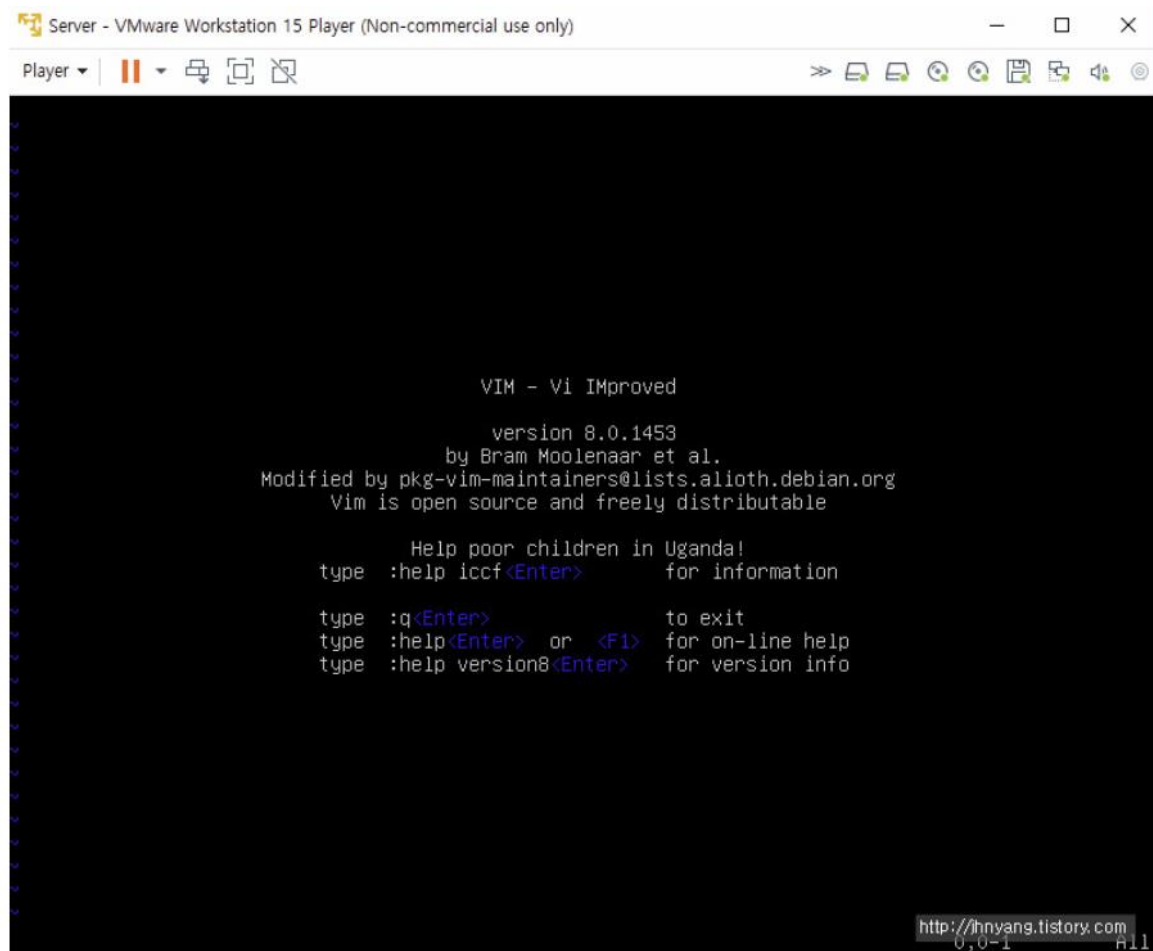
"test.cpp" 9L, 117C 저장 했습니다 9,0-1 모두
```

VI 편집기

- Vim
 - Vi improved의 약자
 - vi가 개선된 버전
 - 사용자 인터페이스, 강력한 문법 강조 기능, 다중 창 및 다중 버퍼 기능 등을 제공
 - 다양한 플러그인과 확장 기능을 제공

VI 편집기

- vi 시작
 - 터미널 창에서 vi를 입력하면 vi 에디터가 실행됨



```
Server - VMware Workstation 15 Player (Non-commercial use only)
Player ▾ || ▾ [ ] [ ] [ ]
VIM - Vi IMproved
      version 8.0.1453
      by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Vim is open source and freely distributable

  Help poor children in Uganda!
type  :help iccf<Enter>      for information

type  :q<Enter>              to exit
type  :help<Enter> or <F1>   for on-line help
type  :help version8<Enter> for version info

http://hnyang.tistory.com
0,0-1 H11
```

VI 편집기

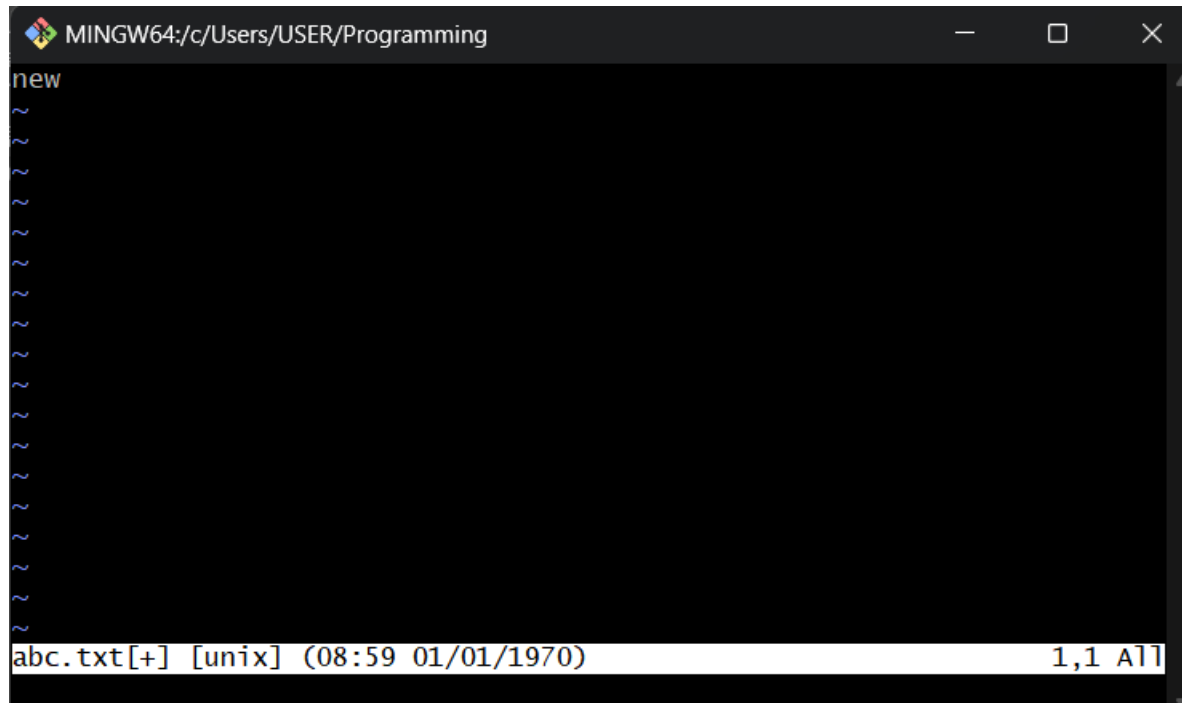
- 새 파일 열어 vi 시작
 - `vi new.txt`
 - 터미널 창에서 위와 같이 입력하면 빈 화면이 열리고 새로운 파일인 "new.txt"를 편집할 수 있게 됨
 - 만약 new.txt가 이미 있다면 이미 있는 파일에서 편집하게 됨

VI 편집기

- vi 편집기 모드
 - vi 편집기는 세 가지 모드를 제공함
 - 명령모드, 입력모드, ex 명령모드
 - 이 모드들 사이 전환하면서 텍스트 파일을 편집함

VI 편집기

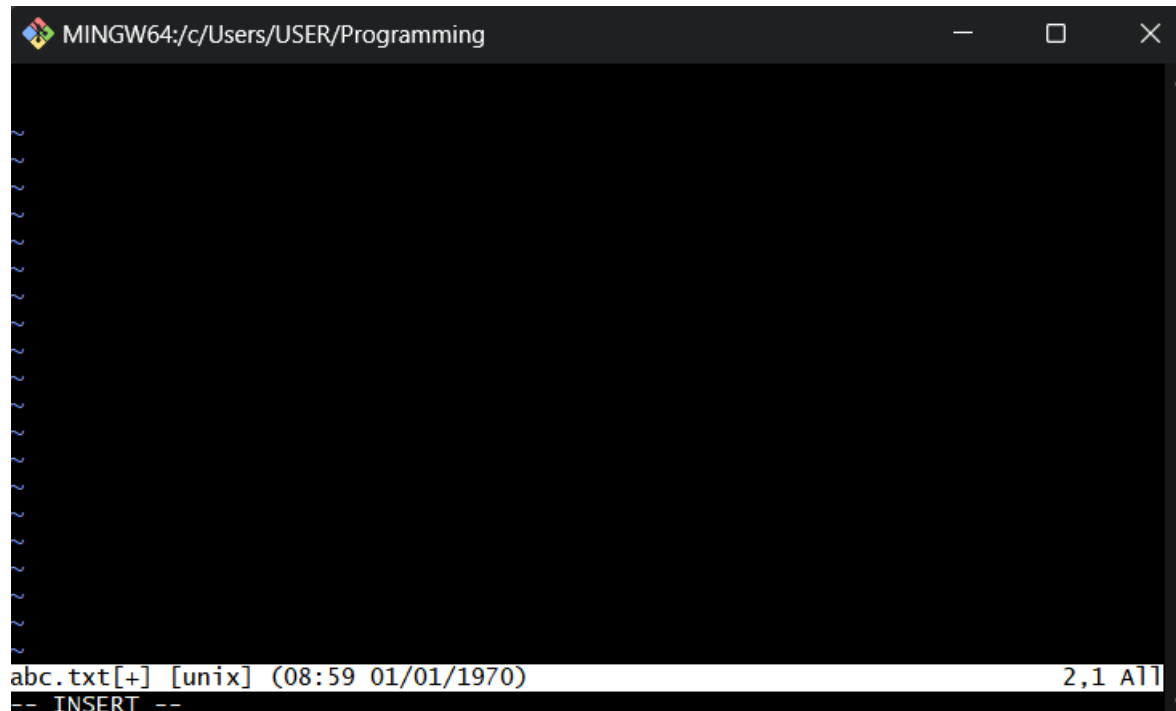
- 명령모드
 - vi를 실행시키면 가장 먼저 접하게 되는 기본이 되는 모드
 - 커서의 이동, 수정, 삭제, 복사, 붙이기, 탐색 등을 수행
 - 입력 전환키인 i,a,o,l,A,O 등을 입력하면 입력 모드로 전환됨
 - 다른 모드에서 명령 모드로 다시 전환하려면 [Esc]를 누르면 됨
 - 맨 아래에 아무것도 안 써있으면 명령모드



The screenshot shows a terminal window titled "MINGW64:/c/Users/USER/Programming". Inside the terminal, the vi editor is open with a file named "new". The editor is in command mode, indicated by the "new" prompt at the top left. The main area of the editor is black with several blue tilde (~) characters on the left side, representing line numbers. At the bottom of the terminal, a status bar shows "abc.txt[+] [unix] (08:59 01/01/1970)" on the left and "1,1 A" on the right.

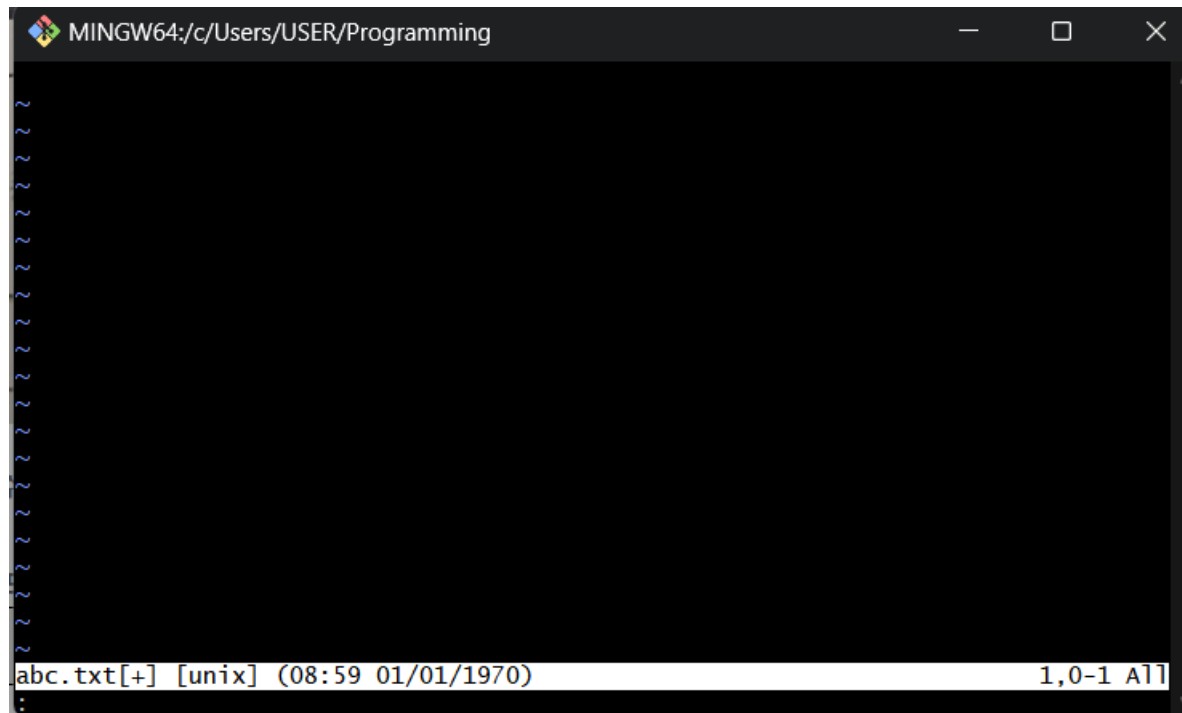
VI 편집기

- 입력모드
 - 입력모드, 편집모드, input mode, insert mode 등으로 불림
 - 글자를 입력하면서 문서를 작성하는 모드
 - 명령 모드에서 입력 전환키를 눌러 전환하면 화면 아래에 "-- INSERT--"라고 표시되며 입력모드로 전환됨



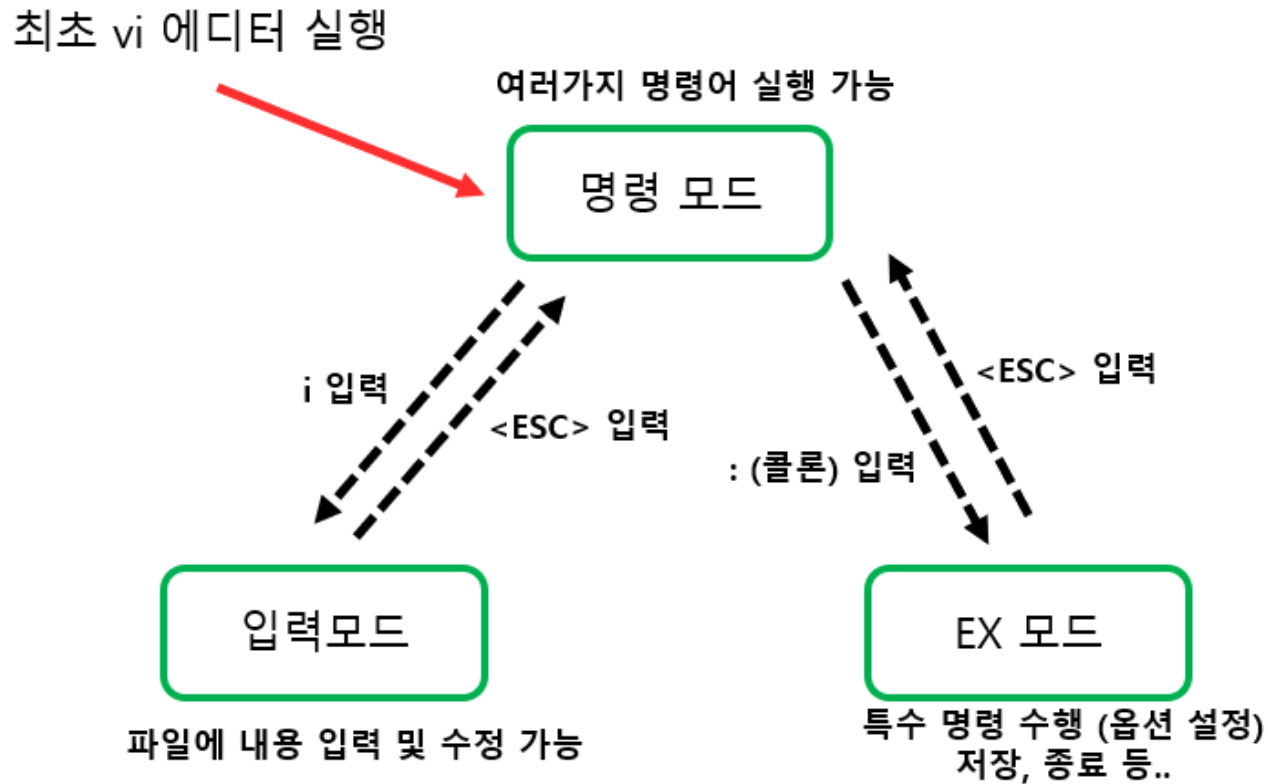
VI 편집기

- ex 명령모드
 - 명령모드에서 ":" 키를 입력했을 때 화면 맨 아랫줄에서 ":"가 표시되며 명령을 수행할 수 있는 모드
 - 저장, 종료, 탐색, 치환 및 vi 환경 설정 등의 역할을 하는 모드



VI 편집기

- 모드전환



VI 편집기

- 입력모드로의 전환
 - 명령모드에서 i,a,o,l,A,O 중 하나를 입력하면 입력모드로 들어갈 수 있음
 - 다음과 같은 차이점이 있음

i	현재 커서의 위치부터 입력	l	현재 커서 줄의 맨 앞에서부터 입력
a	현재 커서의 위치 다음 칸부터 입력	A	현재 커서 줄의 맨 마지막부터 입력
o	현재 커서의 다음 줄에 입력	O	현재 커서의 이전 줄에 입력
s	현재 커서 위치의 한 글자를 지우고 입력	S	현재 커서의 한 줄을 지우고 입력

VI 편집기 명령모드/ex 명령모드

- 방향키
 - 기본 방향키 뿐 아니라 아래 키 입력도 방향키 역할을 함
 - h
 - 왼쪽
 - j
 - 아래
 - k
 - 위
 - l
 - 오른쪽

VI 편집기 명령모드/ex 명령모드

- 단어 단위 이동
 - 단어 단위로 이동할 수 있음
 - w
 - 다음 단어의 첫 글자로 이동
 - b
 - 이전 단어의 첫 글자로 이동
- 행의 첫/마지막
 - 같은 행에서 제일 처음 글자 및 마지막 글자로 이동 가능
 - ^
 - 그 행 처음 글자로 이동
 - \$
 - 그 행 마지막 글자로 이동

VI 편집기 명령모드/ex 명령모드

- 이전/다음 행 이동
 - +
 - 다음 행의 첫 글자로 이동
 - -
 - 이전 행의 첫 글자로 이동
- 문서의 시작/끝
 - gg
 - 문서의 맨 첫 행으로 이동
 - G
 - 문서의 맨 마지막 행으로 이동

VI 편집기 명령모드/ex 명령모드

- 임의 행으로 이동
 - :[n]
 - n번째 행으로 이동. [n]은 임의의 번호
 - ex) :10
- 문단 단위 이동
 - {
 - 이전 문단으로 이동
 - }
 - 다음 문단으로 이동

VI 편집기 명령모드/ex 명령모드

- 화면 이동
 - Ctrl+b
 - 이전 화면으로 이동
 - Page Up에 해당
 - Ctrl+F
 - 다음 화면으로 이동
 - Page Down에 해당

VI 편집기 명령모드/ex 명령모드

- 기본 규칙

- 명령 앞에 숫자를 넣으면 그 명령 앞에 누른 숫자만큼 반복한다는 의미가 있음
- ex) + 명령어가 다음 행으로 이동하는 것이라면 3+ 명령어는 세 번째 다음 행으로 이동하는 것이 됨

VI 편집기 명령모드/ex 명령모드

- 삭제
 - x
 - 커서 위치의 글자 삭제
 - nx
 - n개의 글자 삭제
 - dw
 - 한 단어를 삭제
 - ndw
 - n개의 단어를 삭제
 - dd
 - 한 행을 삭제
 - ndd
 - n개의 행을 삭제
 - D
 - 커서 위치부터 행의 끝까지 삭제

VI 편집기 명령모드/ex 명령모드

- 복사
 - :%y
 - 문서 전체를 복사함
 - :a,by
 - a~b행을 복사함
 - ex) :1,5y 명령어는 1~5행을 복사하라는 의미
 - yw
 - 현재 커서 위치의 한 단어를 복사
 - nyw
 - 현재 커서 위치의 n개의 단어를 복사
 - yy
 - 현재 커서 위치의 한 행을 복사
 - nyy
 - 현재 커서 위치의 n개의 행을 복사

VI 편집기 명령모드/ex 명령모드

- 붙여넣기
 - p
 - 복사한 내용을 현재 행 이후에 붙여넣기
 - P
 - 복사한 내용을 현재 행 이전에 붙여넣기

VI 편집기 명령모드/ex 명령모드

- 블록지정
 - 원하는 위치에서 v를 누르고 커서를 이동시키면 블록이 형성됨
 - 블록을 지정한 후 지우려면 d, 복사하려면 y를 누르면 됨
- 되돌리기
 - u
 - 직전에 내린 명령을 취소함

VI 편집기 명령모드/ex 명령모드

- 저장
 - :w
 - 저장
 - :w file.txt
 - file.txt라는 이름으로 파일 저장
- 종료
 - :q
 - vi 종료
 - :q!
 - 저장하지 않고 종료
 - :wq!
 - 강제 저장 후 종료

VI 편집기 명령모드/ex 명령모드

- 탐색
 - `:[찾을문자열]`
 - 현재 커서 아래방향으로 탐색
 - ex) `:/text`
 - `:?[찾을문자열]`
 - 현재 커서 윗방향으로 탐색
 - ex) `:?text`
 - `n`
 - 다음 문자열 탐색
 - `N`
 - 이전 문자열 탐색

커널

- 운영체제의 핵심 부분
- 하드웨어와 소프트웨어를 연결하여 서로 상호작용할 수 있도록 도움
- 이를 통해 사용자가 컴퓨터를 조작하고 다양한 SW 실행 가능
- 하드웨어 관리, 프로세스 관리, 메모리 관리, 파일 시스템 관리, 네트워크 관리 등 다양한 관리를 하며 안정적으로 시스템을 운영

Shell이란?

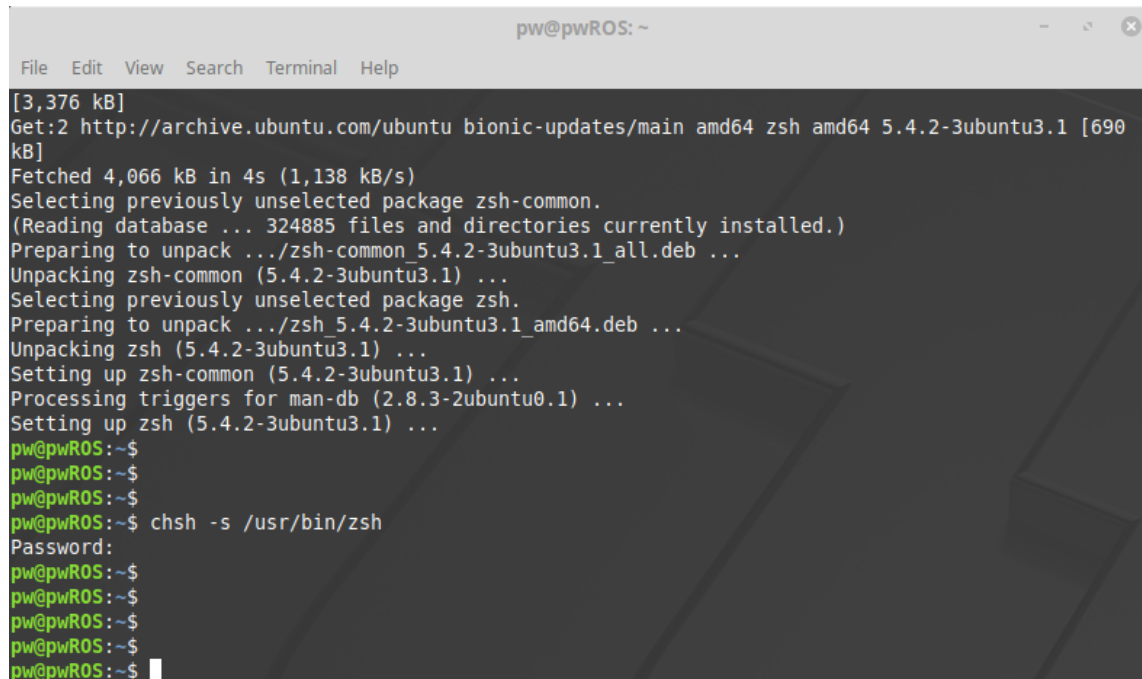
- 커널과 사용자 사이를 이어주는 역할을 하는 명령어 처리기
- 사용자로부터 명령어를 입력 받아 명령어를 처리

Shell 종류

- Bourne Shell
 - sh
- C Shell
 - csh
- tee-see-Shell
 - tcsh
- Z shell
 - zsh
- Bourne-again shell
 - bash
 - 현 시대에 가장 많이 쓰이는 shell
 - 리눅스, 맥 OS 등의 기본 shell로 채택됨

터미널

- 터미널이란?
 - 터미널은 컴퓨터에서 명령어를 입력하고 실행하는 인터페이스
 - GUI가 없는 환경에서 사용되며 텍스트로만 구성된 화면으로 명령어를 입력하고 실행결과를 확인
 - 리눅스에서 터미널은 기본적으로 제공되는 shell을 사용



```
pw@pwROS: ~  
File Edit View Search Terminal Help  
[3,376 kB]  
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 zsh amd64 5.4.2-3ubuntu3.1 [690 kB]  
Fetched 4,066 kB in 4s (1,138 kB/s)  
Selecting previously unselected package zsh-common.  
(Reading database ... 324885 files and directories currently installed.)  
Preparing to unpack .../zsh-common_5.4.2-3ubuntu3.1_all.deb ...  
Unpacking zsh-common (5.4.2-3ubuntu3.1) ...  
Selecting previously unselected package zsh.  
Preparing to unpack .../zsh_5.4.2-3ubuntu3.1_amd64.deb ...  
Unpacking zsh (5.4.2-3ubuntu3.1) ...  
Setting up zsh-common (5.4.2-3ubuntu3.1) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
Setting up zsh (5.4.2-3ubuntu3.1) ...  
pw@pwROS:~$  
pw@pwROS:~$  
pw@pwROS:~$  
pw@pwROS:~$ chsh -s /usr/bin/zsh  
Password:  
pw@pwROS:~$  
pw@pwROS:~$  
pw@pwROS:~$  
pw@pwROS:~$  
pw@pwROS:~$
```

Shell과 터미널 관계

- Shell
 - Shell은 레스토랑의 셰프와 같음
 - 터미널에 입력한 명령을 처리하는 명령줄 인터프리터임
 - 터미널에 명령을 입력하면 shell이 대신 해당 명령을 읽고 해석하고 실행
- 터미널
 - 터미널은 레스토랑의 웨어터와 같음
 - 텍스트 명령을 입력하여 linux 운영체제와 상호작용할 수 있는 사용자 인터페이스임
 - 터미널은 명령을 받아 처리할 shell(셰프)에 전달함
 - 그 후 shell은 결과 또는 출력을 반환하며 터미널은 이를 사용자에게 다시 표시함

경로 이동

- `cd <디렉토리이름>`
 - 위 명령어를 통해 디렉토리로 이동 가능
 - ex) 현재 디렉토리에 `GitDirectory1`이라는 디렉토리가 있다고 할 때 "`cd GitDirectory1`" 명령을 통해 해당 디렉토리로 들어감
 - 절대 경로나 상대 경로를 적어주면 현재 디렉토리 뿐 아니라 임의의 위치로 이동 가능

경로

- /
 - 최상위 디렉토리 혹은 루트 디렉토리
 - ex) "cd /" 을 입력하면 루트 디렉토리로 이동
- ~
 - 홈 디렉토리임
 - ex) "cd ~"을 입력하면 홈 디렉토리로 이동함
 - 홈 디렉토리의 실제 경로는 /home/eslee3209 같은 형태일 수 있음 (eslee3209은 계정 이름)

경로

- ..
 - 상위 디렉토리를 의미
 - ex) "cd .." 명령어를 입력하면 상위 디렉토리로 이동
- .
 - 현재 디렉토리를 의미
 - ex) "cd ." 명령어를 입력하면 그대로 있음

절대경로

- 최상위 디렉토리(/)부터 시작해서 목표 디렉토리까지 가는 경로를 전부 기술하는 방식
- 절대경로를 기술할 때 항상 맨 앞에 최상위 디렉토리 (/)가 붙음
- ex) `cd /home/eslee3209/GitDirectory1`

상대경로

- 현재 자신이 있는 위치를 기준으로 이동
- 예를 들어, 현재 위치에서 GitDirectory1 디렉토리에 들어간 후 temp 디렉토리에 들어간다고 하면 다음 명령어로 가능
 - `cd ./GitDirectory1/temp`
- `..`을 사용하여 이전 디렉토리로도 갈 수 있음
 - `"cd .."`은 이전 디렉토리로 이동
 - `"cd ../../"`을 입력하면 이전, 이전 디렉토리로 이동

현재 경로

- pwd
 - 현재 디렉토리의 위치를 표시해주는 명령어
 - Present Working Directory의 약자

echo

- echo는 메아리라는 뜻으로 문자열을 표시할 수 있는 명령어
- 다음 형태로 문자열을 화면에 표시함
 - echo <문자열>
 - ex) echo "Hello Git"

cat

- 파일의 내용을 쭉 화면에 뿌려줌
- ex) cat file1.txt

```
for@Eunsang MINGW64 ~/Programming/hello-git-cli (master)
$ cat file1.txt
hello git
```

명령어 히스토리

- 터미널에서 위쪽 화살표를 누르면 이전에 수행했던 명령어들을 최신 순서로 쭉 볼 수 있음
- 즉, 터미널에서 수행한 명령어 히스토리 확인 가능
- history
 - 위 명령어를 터미널에 입력하면 최근 많은 명령어 리스트가 보임

디렉토리 생성/삭제

- 디렉토리 생성
 - mkdir <새 디렉토리 이름>
 - 위 명령어를 통해 새 디렉토리를 생성할 수 있음. rm은 remove의 약자.
 - ex) mkdir GitDirectory1
- 디렉토리 삭제
 - rm -r <디렉토리 이름>
 - 위 명령어를 통해 이미 있는 디렉토리를 삭제할 수 있음
 - ex) rm -r GitDirectory1

디렉토리 복사/이름바꾸기

- 디렉토리 복사
 - `cp -r directory1 directory2`
 - `directory1` 디렉토리를 `directory2`라는 이름으로 복사
- 디렉토리 이름바꾸기
 - `mv directory1 directory2`
 - 혹은 `mv ./directory1 ./directory2`
 - `directory1` 디렉토리의 이름을 `directory2`로 바꿈

파일 복사

- `cp [file1] [file2]`
 - file1 파일을 file2라는 이름으로 복사
- `cp file1 dir1/`
 - file1을 dir1 디렉토리 안에 복사
- `cp *.txt dir1/`
 - *.txt의 의미는 txt 확장자의 모든 파일을 일괄적으로 복사한다는 의미
 - 모든 txt 파일들을 dir1 디렉토리로 복사

파일 삭제

- `rm <파일명>`
- 위 명령어로 이미 있는 파일을 삭제할 수 있음
- ex) `rm GitFile1.txt`

파일 이름바꾸기/이동

- `mv [file1] [file2]`
 - `mv`는 `move`의 약자임
 - `file1`의 이름을 `file2`로 바꿈
 - ex) `mv GitFile1.txt GitFile2.txt`
- `mv [file1] [directory1]`
 - `file1`이 디렉토리 `directory1`으로 이동됨
 - ex) `mv GitFile1.txt temp` (`temp`라는 디렉토리가 있다고 가정)

파일이름 자동완성

- 파일이름을 입력할 때 일부만 입력하고 tab을 눌러주면 파일 이름이 완성됨
- 예를 들어 VeryVeryLongNameFile.txt라는 파일이 현재 디렉토리에 있다고 가정
- "Very"까지만 입력하고 tab을 누르면 전체 파일 이름 "VeryVeryLongNameFile.txt"이 완성됨

현재 디렉토리 상태

- `ls`
 - list의 약자
 - 현재 디렉토리에 있는 파일/디렉토리를 볼 수 있음
- `ls -a`
 - 숨김 파일까지 볼 수 있음
- `ls -al [디렉토리명]`
 - 해당 디렉토리 내부를 확인
 - ex) `ls -al .git`

```
$ ls -al .git
total 11
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 ./
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 ../
-rw-r--r-- 1 for 197121 23 Feb 13 16:59 HEAD
-rw-r--r-- 1 for 197121 130 Feb 13 16:59 config
-rw-r--r-- 1 for 197121  73 Feb 13 16:59 description
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 hooks/
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 info/
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 objects/
drwxr-xr-x 1 for 197121  0 Feb 13 16:59 refs/
```

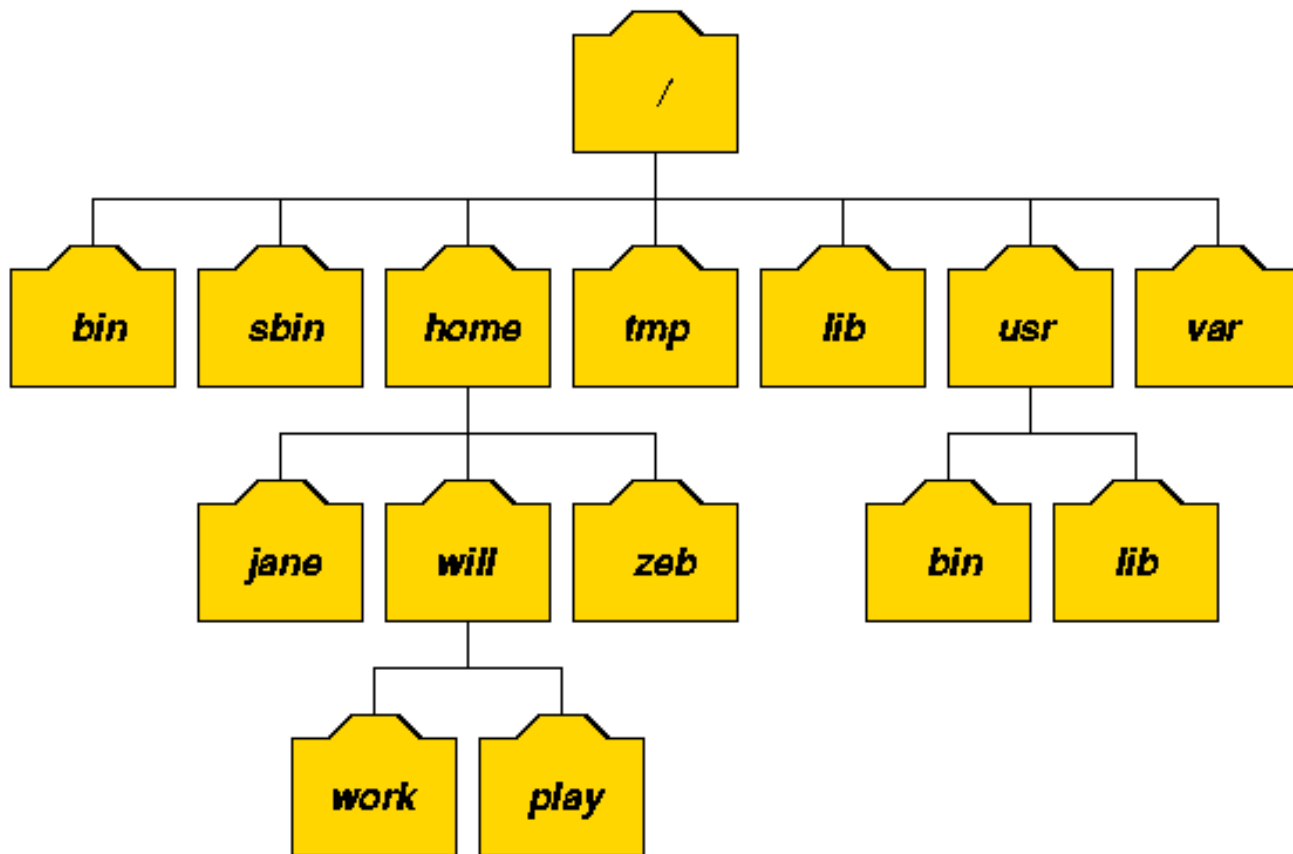
다운로드

- wget [웹사이트]
 - 해당 웹사이트 링크의 단일 파일을 다운로드 받음
 - ex) wget <http://www.openss7.org/repos/tarballs/strx25-0.9.2.1.tar.bz2>

디렉토리 구조

- 리눅스의 디렉토리 혹은 파일 시스템 구조는 윈도우와 조금 다른 구조를 가짐
- 기본적으로 디렉토리를 구분하는 '/'(슬래시)는 리눅스에서 사용하고 윈도우는 반대인 '\\'(역슬래시)를 사용
- 디렉토리 명칭 또한 리눅스에서는 디렉토리(directory), 윈도우에서는 폴더(folder)라고 부름

디렉토리 구조



디렉토리 구조

- 예시

```
eslee3209@ubuntu:/$ ls
bin      etc          jupyterhub.sqlite  libx32      NVIDIA-Linux  pretrained_parameters  rstudio-server-1.3.1073-amd64.deb  srv      usr
boot     home         lib                lost+found  opt           proc                 run                                sys      var
data     jupyterhub_cookie_secret  lib32         media       path          r_jupyterhub.Rout     sbin                             testFile
dev      jupyterhub-proxy.pid     lib64         mnt        PATH          root                 snap                             tmp
```

- 루트(/)에 들어가보면 bin, sbin, home, tmp, lib, usr, var 디렉토리가 실제 있음

디렉토리 구조

- 리눅스 시스템 디렉토리 구조는 전체적으로 역 트리 (tree) 구조임
- 명령어의 종류와 성격, 사용권한에 따라 각각의 디렉토리들로 구분됨
- 리눅스 배포판들은 '리눅스 파일시스템 표준'인 FSSTND(Linux File System Standard) 라는 표준을 준수하므로 대부분의 리눅스 배포판들은 그 기본 골격이 같음

디렉토리 구조

- /(루트)
 - 최상의 디렉토리인 루트 디렉토리를 의미함
 - 리눅스의 모든 디렉토리들의 시작점
 - 즉, 모든 디렉토리들을 절대경로로 표기할 때에 이 디렉토리로부터 시작해야함

디렉토리 구조

- /bin
 - 기본적인 명령어가 저장된 디렉토리
 - 즉, 리눅스 시스템 사용에 있어 가장 기본적이라고 할 수 있는 mv, cp, rm 등과 같은 명령어들이 이 디렉토리에 존재하며 root 사용자와 일반사용자가 함께 사용할 수 있는 명령어 디렉토리임

```
foo2hiperc-wrapper  lzmainfo            python               xzfgrep
foo2hp              lzmore              python2             xzgrep
foo2hp2600-wrapper m2300w              python2.7           xzless
foo2lava            m2300w-wrapper      python3             xzmore
foo2lava-wrapper   m2400w              python3.8           y2racc2.7
foo2oak             m4                  python3.8-config    yacc
foo2oak-wrapper    macptopbm           python3-config       yaml2obj
foo2qpd             mag                 python3-futurize     yaml2obj-10
foo2qpd-wrapper    make                python3-pasteurize   yaml-bench-10
foo2slx             makeconv            pyversions          yarn
foo2xqx             makedtx             qpdldcode           ybmtopbm
foo2xqx             make-first-existing-target qrttoppm            yelp
foo2xqx-wrapper    makeglossaries      quickbook           yes
foo2zjs             makeglossaries-lite quirks-handler       ypdomainname
foo2zjs-icc2ps      makeindex            R                   yplan
foo2zjs-pstops      makeinfo             racc2.7             yuvsplittoppm
foo2zjs-wrapper     makejvf              racc2y2.7           yuvtoppm
foomatic-rip        mako-render          rake                 zcat
fprintd-delete      man                   raku                 zcmp
fprintd-enroll      mandb                 raku-debug           zdiff
fprintd-list        manpath               rakudo               zdump
fprintd-verify      man-recode            rakudo-debug         zegrep
free                 mapscrn               rakudo-debug-m       zeisstopnm
from                 mathspic              rakudo-gdb-m         zenity
fst2vcd             mattrib               rakudo-lldb-m        zfgrep
fstminer            mawk                  rakudo-m             zforce
fstopgm             mbadblocks            rakudo-valgrind-m    zgrep
ftp                 mcat                  ranlib               zip
funzip              mcd                   rasttopnm            zipcloak
fuser               mcheck                rawtopgm             zipdetails
fusermount          mclasserace          rawtoppm             zipgrep
futurize            mcomp                 rbash                zipinfo
fwupdagent          mcookie               rcp                  zipnote
fwupdate            mcopy                 rctest               zipsplit
fwupdmgr            md5sum                rdifffdir            zjsdecode
fwupdtool           md5sum.textutils     rdjpgcom             zless
fwupdtmpevlog       mdatopbm              rdma                  zmore
g++                 mdel                  rdoc                  znew
g3topbm             mdeltree              rdoc2.7
```

eslee3209@ubuntu:/bin\$ |

디렉토리 구조

- /boot
 - 리눅스 부트로더(Boot Loader)가 존재하는 디렉토리
 - 즉, GRUB과 같은 부트로더에 관한 파일들이 이 디렉토리에 존재함
- /dev
 - 시스템 디바이스(device)파일을 저장하고 있는 디렉토리
 - 즉, 하드디스크 장치파일 /dev/sda, CD-ROM 장치파일 /dev/cdrom 등과 같은 장치파일들이 존재하는 디렉토리

디렉토리 구조

- /etc
 - 시스템의 거의 모든 설정파일들이 존재하는 디렉토리
 - /etc/sysconfig(시스템 제어판용 설정파일), /etc/passwd(사용자 관리 설정파일) 등과 같은 파일들이 존재
- /home
 - 사용자의 홈 디렉토리
 - useradd 명령어로 새로운 사용자를 생성하면 대부분 사용자의 ID와 동일한 이름의 디렉토리가 자동으로 생성됨

```
eslee3209@ubuntu:/home$ ls
bravokoo  chae950104  cryptoldh  dasan  eslee3209  jhlee  joonwoo3511  kkt1513  sjpark  sonic  sonyongyi
```

디렉토리 구조

- /lib
 - 커널모듈파일과 라이브러리파일, 즉, 커널이 필요로하는 커널모듈파일들과 프로그램(C, C++ 등)에 필요한 각종 라이브러리 파일들이 존재하는 디렉토리
- /root
 - 시스템 최고 관리자인 root 사용자의 개인 홈디렉토리
- /tmp
 - 일명 "공용디렉토리"
 - 시스템을 사용하는 모든 사용자들이 공동으로 사용하는 디렉토리

디렉토리 구조

- /proc
 - 일명 "가상파일시스템"이라고 하는 곳
 - 현재 메모리에 존재하는 모든 작업들이 파일형태로 존재하는 곳
 - 디스크상에 실제 존재하는 것이 아니라 메모리 상에 존재하기 때문에 가상파일시스템이라고 부름
 - 실제 운용상태를 정확하게 파악할 수 있는 중요한 정보를 제공하며 여기에 존재하는 파일들 가운데 현재 실행중인 커널(kernel)의 옵션 값을 즉시 변경할 수 있는 파라미터파일들이 있기 때문에 시스템 운용에 있어 매우 중요한 의미를 가짐
- /usr
 - 시스템이 아닌 일반사용자들이 주로 사용하는 디렉토리
 - c++, cpp, crontab, du, find 등과 같이 일반사용자들용 명령어들은 /usr/bin에 위치함