

Vim에서의 LATEX 환경 설정 및 사용법

1 들어가며

제가 사용하는 LATEX 환경설정에 대한 파일입니다. 엄청 간단하고 미니멀합니다. 그런데 작업은 무지 빨라지고 편해집니다. 공유를 하면 참 좋겠다고 평소에 생각을 많이 했었습니다. `.vimrc` 설정이나, `.sty` 파일 등이 너무 배배 꼬여서 보여주기 부끄러운 상태라 여태껏 공개를 안 했습니다. 이번 여유가 있을 때 정리를 해서 메뉴얼까지 배포하려고 합니다. 부족한 글 솜씨지만 넓은 아량으로 이해해주시면 감사하겠습니다.

제작자 컴퓨터 환경

저는 Ubuntu 16.04 LTS (Gnome) 환경에서 작업을 합니다. 컴파일은 `pdflatex` 명령어를 통하여 합니다. 메뉴얼의 모든 작업은 리눅스 환경에서 이루어집니다. 죄송스럽지만, 리눅스 환경이 아니라면 적당히 메뉴얼을 잘 걸러 해석해주셔야 할 것입니다.. (윈도우 버전은 다음에 시간이 되면 업데이트하겠습니다.)

2 Vim 설정

설치는 간단하게 넘어가도록 하겠습니다. Ubuntu 환경에서는 다음과 같이 설치할 수 있습니다. 다음 코드를 터미널에 입력하시면 됩니다. 윈도우나 OS X에서의 환경은 다른 웹페이지에서 따로 참고하시길 바랍니다.

```
$ sudo apt-get update  
$ sudo apt-get install vim
```

우분투의 경우 `vim`이 아닌 `vim-tiny` 버전으로 이미 설치가 되어있을 가능성이 있습니다. 이러할 경우

```
$ sudo apt-get purge  
$ sudo apt-get install vim
```

으로 재설치를 하시길 바랍니다.

터미널에서 `vim`이라고 입력하신 뒤 엔터를 누르시면 바로 실행시킬 수 있습니다.

Vim에서는 여러 가지 플러그인들을 깔면 매우 편합니다. LATEX 문서를 작업할 때는 특히

- YouCompleteMe (리눅스, OS X 환경에서만 설치 권장)

를 추천드립니다. Latex-Suite 등 다른 유명한 플러그인들도 있다만, 한글로 문서를 작업할 때는 영 좋지 못한 것 같습니다. 그래서 LATEX에 직접적으로 연관된 플러그인들의 설치는 지양하도록 합시다.

플러그인 설치 및 `.vimrc` 파일 설정 방법은 Github 포스팅을 참고하시길 바랍니다.

.latexrc 설정

LATEX을 쓰는데 편리한 단축키 소스들을 `.latexrc`으로 작성하여 첨부하였습니다. 이를 `$HOME/.vim/.latexrc` 위치에 집어넣고 아래 코드를 `.vimrc` 파일의 맨 아래쪽에 집어넣어 줍시다.

```
source $HOME/.vim/.latexrc  
au BufRead *.latexrc set filetype=vim
```

.latexrc 파일에 있는 내용은 메뉴얼의 뒷 부분에서 다 설명드리도록 하겠습니다. 모두 다 결국 nmap 과 imap으로만 이루어진 간단한 코드임을 확인하실 수 있을 것입니다. 각자 자주 사용하는 방식대로 임의로 수정하셔도 괜찮습니다.

3 스타일 파일 설정

LATEX에서 sty 파일은 C의 헤더파일, 자바나 파이썬의 다른 클래스 파일과 비슷하다고 보시면 됩니다. 예를 들어 보겠습니다. `woosung.sty` 파일에 다음과 같이 입력하여 저장했다고 합시다.

```
\usepackage{setspace}
\usepackage{kotex}
\newcommand{\hw}{Hello World!}
```

같은 폴더 안에 tex 파일을 만들어 다음과 같이 입력하고 컴파일하면

```
\documentclass{article}
\usepackage{woosung}
\begin{document}
\setstretch{1.25}
안녕하세요! 한글이 입력이 되네! \hw
\end{document}
```

아마 다음과 같은 내용의 pdf 문서가 나올 것입니다.

안녕하세요! 한글이 입력이 되네! Hello World!

원래 `\setstretch` 명령어를 사용하거나, 한글 문서를 조판하려면 `setspace`, `kotex` 패키지를 각각 추가해 줘야합니다. 스타일 파일에 이 패키지를 불러오는 코드를 적고, 이 스타일 파일을 `\usepackage` 명령어로 불러옴으로써 함께 로드가 된 것입니다. 이와 같이 자주 사용하게 될 긴 명령어들의 목록은 sty 파일로 만들어 미리 저장해두는 것을 권장합니다.

전역적으로 접근할 수 있게 만들기

매우 자주 쓰는 sty 파일들은 어디에서나 접근할 수 있도록 따로 처리해줄 수 있습니다. 터미널에서 다음과 같은 명령어를 입력하세요.

```
$ cd $HOME && mkdir .texmf && mkdir .texmf/tex && mkdir .texmf/tex/latex
$ kpsewhich texmf.cnf
```

두 번째 줄의 명령어로 나온 경로에 있는 파일을 아래 명령어로 여세요. 가령, `/etc/texmf/web2c/texmf.cnf`로 나왔다고 하면, 이렇게 치시면 됩니다.

```
$ gedit /etc/texmf/web2c/texmf.cnf
```

파일이 열리면 `TEXMFHOME`이라는 항목을 찾아서 이렇게 바꿔줍니다.

```
$ TEXMFHOME = ~/.texmf
```

이제 `$HOME/.texmf/tex/latex` 폴더 안에 자주 쓰는 sty 파일을 집어넣으세요. 그런 뒤, 다시 터미널로 돌아와서 아래와 같은 명령어를 실행시켜주세요.

```
$ sudo texhash
```

이제부터 따로 해당 sty 파일을 같은 디렉토리에 넣을 필요 없이 어디서든 접근 가능하게 됩니다.

Woosung LATEX 스타일 파일

첨부된 `woosung_hw.sty`, `woosung_korean.sty`, `woosung_math.sty` 파일은 제가 유용하게 사용하는 스타일 파일들입니다. 이것도 매우 미니멀합니다.

- `woosung_hw.sty`: 숙제 파일을 만들 때 주로 사용하던 양식인데, 실험 리포트를 비롯한 대부분의 문서에서 이 스타일을 불러와 사용합니다.
- `woosung_korean.sty`: 한글로 문서를 조판할 때 불러와 사용합니다. `\setstretch`로 문장간 간격에 대한 조정이 되어있습니다.
- `woosung_math.sty`: 수식을 입력할 때 매우 편하게 사용하기 위해 만들었습니다. Theorem, Definition 등을 입력해야 할 경우 유용합니다.

특히, `woosung_math.sty` 파일에 있는 command들이 매우 편리합니다. 파일을 열어보면 다음과 같은 코드가 있습니다.

```
...
\newcommand{\R}{\mathbb{R}}
\newcommand{\N}{\mathbb{N}}
\newcommand{\Z}{\mathbb{Z}}
% Analysis
\newcommand{\ep}{\epsilon}
\newcommand{\dt}{\delta}
\renewcommand{\liminf}[1]{\lim_{#1} \rightarrow \infty} % Excluded when using liminf
\newcommand{\limninf}[1]{\lim_{#1} \rightarrow -\infty}
...
...
```

만약 어떤 문서에서 `\usepackage{woosung_math}` 를 한 다음 아래와 같은 수식을 입력하면

```
\forall n \in \N, \liminf{c} cn = \infty
```

아래와 동일한 효과를 얻습니다.

```
\forall n \in \mathbb{N}, \lim_n \rightarrow \infty cn = \infty
```

즉, 이런 내용의 수식이 입력됩니다.

$$\forall n \in \mathbb{N}, \lim_{c \rightarrow \infty} cn = \infty$$

이와 같이 좀 더 단순하게 수학 기호를 입력할 수 있습니다. 대표적인 것들을 몇 개 꼽자면,

- `\ra`: \rightarrow
- `\la`: \leftarrow
- `\Ra`: \Rightarrow
- `\La`: \Leftarrow
- `\R`, `\N`, `\Z`: $(\mathbb{R}, \mathbb{N}, \mathbb{Z})$
- `\ep`, `\dt`: (ϵ, δ)
- `\inv`: $^{-1}$
- `\ie`: $(i.e.,)$
- `\eg`: $(e.g.,)$

가 있습니다. 또한, `\numberthis` 라는 명령어도 있습니다. 이는 `\begin{align*} ... \end{align*}` 환경 내에서 일부 수식에만 번호를 매길 때 사용합니다. 예를 들어 아래와 같이 입력하면

```
\begin{align*}
\sum_{i=1}^n i &= 1 + 2 + 3 + \cdots + n \\
&\stackrel{\text{\scriptsize \texttt{\backslash numberthis}}}{=} \frac{n(n+1)}{2} \numberthis
\end{align*}
```

이와 같은 수식이 나옵니다.

$$\begin{aligned} \sum_{i=1}^n i &= 1 + 2 + 3 + \cdots + n \\ &= \frac{n(n+1)}{2} \end{aligned} \tag{1}$$

오직 두 번째 줄에서만 `\numberthis` 처리를 해줬기 때문에 수식 번호가 매겨진 것입니다.

4 .latexrc 단축키 설명

자주 사용하는 명령어는 볼드체로 강조해놓았습니다.

컴파일 및 pdf 파일 열기

- [F3] 버튼: pdflatex 컴파일러로 해당 `tex` 파일을 컴파일합니다. 컴파일 로그는 `/tmp/.o.out` 위치에 저장합니다.
- [F4] 버튼: 컴파일된 `pdf` 파일을 default 뷰어로 엽니다.
- [Ctrl+F3] 버튼: gnome-terminal에서 컴파일 로그를 출력하면서 컴파일합니다. [F3]으로 컴파일해서 오류가 생겼을 때 사용하기를 권장드립니다.
- [F5] 버튼: 오류 로그를 오른쪽에 탭으로 띠웁니다.

커서의 이동

‘<<-->>’ 무늬를 찾고 다음 위치로 이동하는 작업으로 편리하게 커서를 이동시킬 수 있습니다. 예를 들어 작업하고 있는 `tex` 파일이 다음과 같다고 가정해봅시다. 커서의 위치는 ‘|’ 표시로 나타내겠습니다.

```
$$ \sum_{i=0}^n <<-->> | $$$
```

여기서 [Ctrl+j] 버튼을 누르면 다음과 같이 커서가 이동합니다.

```
$$ \sum_{i=0}^n <<-->> | $$$
```

‘n’을 입력한 뒤 [Ctrl+j] 버튼을 누르면 커서가 다음과 같이 이동합니다.

```
$$ \sum_{i=0}^n | n | $$$
```

이와 같은 기능으로 키보드만으로도 정말 빠르고 편리하게 조판작업을 할 수 있습니다.

- [Ctrl+j] 버튼: 다음 ‘<<-->>’ 표시로 이동하고 그 문양을 지웁니다.
- [Ctrl+k] 버튼: 커서의 위치에 ‘<<-->>’ 문양을 새로 입력합니다.

문서 정보 입력

- **\Document** 입력: `\begin{document} ... \end{document}` 텍스트를 입력합니다.
- **\Info** 입력: `\title{} \author{} \date{}` 텍스트를 입력합니다.
- **\MoreInfo** 입력: `\title{} \subtitle{} \author{} \date{} \institute{}` 텍스트를 입력합니다. 오직 `beamer` 클래스에서만 적용됩니다.

패키지 불러오기 및 기타

- **\Package** 입력: `\newpackage{}` 텍스트를 입력합니다.
- **\Newcomm** 입력: `\newcommand{}{<<-->}` 텍스트를 입력합니다.
- **\Renewcomm** 입력: `\renewcommand{}{<<-->}` 텍스트를 입력합니다.
(*environment*는 *.sty* 파일에서 미리 처리해놓는 경우가 많기 때문에 따로 단축키로 넣지 않았습니다.)

챕터와 섹션

- **\Chap** 입력: `\chapter{}` 텍스트를 입력합니다.
- **\Sec** 입력: `\section{}` 텍스트를 입력합니다.
- **\SSec** 입력: `\subsection{}` 텍스트를 입력합니다.
- **\SSSec** 입력: `\subsubsection{}` 텍스트를 입력합니다.
- **\Sec*** 입력: `\section*{}` 텍스트를 입력합니다.
 - `\Chap*`, `\SSec*`, `\SSSec*` 도 유사합니다.
 - 챕터 및 섹션에 번호매김을 하지 않습니다.

텍스트 강조

- **\tbf** 입력: 볼드체 텍스트 입력 레이아웃을 적습니다. (*Text*)
- **\tit** 입력: 기울임 텍스트 입력 레이아웃을 적습니다. (*Text*)
- **\ttt** 입력: 모노스페이스 텍스트 입력 레이아웃을 적습니다. (*Text*)
- **\tsc** 입력: Scientific 텍스트 입력 레이아웃을 적습니다. (*TEXT*)
- **\txt** 입력: 플레인 텍스트 입력 레이아웃을 적습니다. 수식 모드에서 플레인 텍스트로 적을 때 주로 사용됩니다.

번호 매김

- **\Enumerate** 입력: 1) 2) 3) ... 순의 번호매김 레이아웃을 적습니다.
 - `\Enumeratei` 입력: i) ii) iii) ... 순
 - `\Enumeratea` 입력: a) b) c) ... 순
 - `\EnumerateI` 입력: i) ii) iii) ... 순 (볼드체)
 - `\EnumerateA` 입력: a) b) c) ... 순 (볼드체)
- **\Itemize** 입력: 번호 없이 리스트하는 레이아웃을 적습니다.

그림, 표, 가운데 정렬

- **\Figure** 입력: 그림을 넣는 레이아웃을 적습니다.

– 아래와 같이 입력이 됩니다.

```
\begin{figure}[!htb]
    \centering
    \includegraphics[width=.1\textwidth]{<<-->}
    \caption{<<-->}
\end{figure}<<-->
```

올바른 입력 예시는 다음과 같습니다.

```
\begin{figure}[!htb]
    \centering
    \includegraphics[width=.8\textwidth]{src/woosung.png}
```

```
\caption{Photo of Woosung Song}
\end{figure}
```

– 커서 이동인 [Ctrl+j] 버튼을 이용하여 차례대로 채우시면 됩니다.

- \Table 입력: 표를 넣는 레이아웃을 적습니다.
- \Tabular 입력: 표와 함께 캡션을 정의할 수 있는 레이아웃을 적습니다. 라벨링이 가능합니다.

수학 관련 단축키

Align 및 Array

- \Align 입력: \begin{align*} ... \end{align*} 를 입력합니다. 한 번에 여러 줄의 수식을 입력할 때 사용합니다. 대부분의 가운데 정렬 수식을 적을 때 이 코드를 사용합니다.
- \Array 입력: 수식 모드 내에서 표 모양 데이터를 입력할 때 사용합니다.

정의, 정리 선언 (woosung_hw.sty 파일 불러올 때만 유효)

- \Thm 입력: \begin{theorem} ... \end{theorem} 을 입력합니다.
 - \Def 입력: \begin{definition} ... \end{definition} 을 입력합니다.
 - \Proof 입력: \begin{proof} ... \end{proof} 을 입력합니다.
- 기타 \Lem, \Prop, \Coro, \Ex, \Prob, \Remark 입력: 각각 lemma, proposition, corollary, example, problem, remark를 의미합니다.

괄호 입력

- \Brace(입력: 소괄호를 좌 우 입력해주고 커서로 쉽게 이동할 수 있도록 레이아웃을 구성해줍니다.
- \Brace{ 입력: 중괄호 입력, 레이아웃 구성해줍니다. 단, \{ .. \} 형태로 구성합니다.
- \Brace[입력: 대괄호 입력, 레이아웃 구성해줍니다.
- \Brace. 입력: 왼쪽은 중괄호를, 오른쪽은 온점을 입력합니다. 아래와 같은 수식을 만들 때 array와 함께 주로 사용됩니다.

$$X = \begin{cases} f(x) & \text{if } x \geq 0 \\ g(x) & \text{otherwise} \end{cases}$$

- \Bracec 입력: ceil 함수 기호에 대응되는 괄호를 양 옆에 입력합니다.
- \Bracef 입력: floor 함수 기호에 대응되는 괄호를 양 옆에 입력합니다.

기타 수학기호 단축키

- \frac{ 입력: 분수 입력 레이아웃을 완성합니다. \frac{|}{|} {<<-->>} <<-->>
- \sum_ 입력: 합 입력 레이아웃을 완성합니다. \sum_{|}^{|} {<<-->>} <<-->>
- \prod_ 입력: 곱 입력 레이아웃을 완성합니다. \prod_{|}^{|} {<<-->>} <<-->>
- \lim_ 입력: 극한 입력 레이아웃을 완성합니다. \lim_{|} {<<-->>}
- \bigcup_ 입력: 합집합 입력 레이아웃을 완성합니다. \bigcup_{|} {<<-->>}
- \bigcap_ 입력: 교집합 입력 레이아웃을 완성합니다. \bigcap_{|} {<<-->>}
- \bigcup^ 입력: 합집합 입력 레이아웃을 완성합니다. \bigcup_{|}^{|} {<<-->>} <<-->>
- \bigcap^ 입력: 교집합 입력 레이아웃을 완성합니다. \bigcap_{|}^{|} {<<-->>} <<-->>

코드 입력 (모노스페이스)

- \Verbatim 입력: 코드 입력 레이아웃을 완성합니다.

- 예를 들어 다음과 같이 코드를 채우면 (백슬래시가 띄어진건 컴파일이 안 돼서 그런거라 이해해 주세요..)

```
\begin{Verbatim}[tabsize=4,xleftmargin=2em]
def woosung():
    print("Hello World!")
    print("I like you");
\end{Verbatim}
```

이런 결과가 나옵니다.

```
def woosung():
    print("Hello World!")
    print("I like you");
```

- *\NVerbatim* 입력: 줄 번호도 나오는 코드 입력 레이아웃을 완성합니다.

```
1 def woosung():
2     print("Hello World!")
3     print("I like you");
```

Beamer 설정

LATEX으로 강의 슬라이드를 만들거나 학회용 포스터를 사용할 때 주로 사용하는 클래스가 `beamer` 입니다. 관련 세팅에 대한 메뉴얼은 추후 추가하도록 하겠습니다.