워드 프로세서 사용자를 위한 LaTeX
version 1.0.10 한국어

Guido Gonzato, Ph.D. guido.gonzato@gmail.com
김강수 (옮김)

2015년 1월 8일　　2015년 8월 12일

요약문

LaTeX 문서 작성은 워드 프로세서에 비해 상당한 장점이 있다. 그러나 초보자들은 그 일을 어떻게 해야 하는지, 필요한 특정 기능을 어디서 찾아야 하는지 알아내기가 어렵다. 이 안내서는 워드 프로세서와 LaTeX 조판을 비교함으로써 워드 프로세서에서 LaTeX으로 이행하는 것을 도와주려 한다. 일반적인 워드 프로세서의 주요 기능을 열거하고 각각이 LaTeX에서 어떻게 실현되는가를 보여준다. 많은 예를 첨부하였다.

역자는 이 문서에 두 가지로 개입하였다. 본문을 충실히 번역하는 이외에 역자의 의견을 방주 형태로 추가하였다. 그리고 일부 소절이나 단락을 추가한 것도 있다. 원칙적으로 원문은 손상하지 않았으며 교정하거나 코멘트할 것이 있으면 모두 위와 같은 형식에 의해서 했다. 본문에 추가한 단락과 소절은 그 사실을 방주에서 밝혔다. 이주호 님께서 문서 전체를 읽고 교정을 보아주셨다.

차 례

표 차례

그림 차례

1 들어가기

시작하기 전에 이 안내서가 LATEX 입문서가 아니라는 점을 말해두고자 한다. 이 문서를 읽고 있다는 것은 적어도 LATEX이 무엇인지, 기본 명령은 어떤 것이 있는지를 알고 있다는 뜻이다. 이 안내서에서 설명하려 하는 것은 LATEX을 사용함으로써 어떻게 워드 프로세서를 효과적으로 대체할 수 있는가 하는 것이다.

워드 프로세서는 오늘날 사무자동화 분야의 '킬러 프로그램'이다. 익숙한 WYSIWYG 인터페이스를 가지고 있기 때문에 LATEX에 비해 더 쉬운 것으로 여겨진다. 대부분의 사무보조원은 꽤 짧은 시간에 그 사용법을 익힌다. 문제는 이 물건이 날이 갈수록 느려지고 비대해지고[1] 버그투성이가 되고 심심하면 죽고 비싸고 바이러스에 감염되고 서로간에 호환불가능하게 되어 간다는 것이다. 그 기본 출력 품질에 대해서는 말하지 않겠다.

LATEX은 훌륭한 대안이다. 어떤 경우 유일한 대안이기도 하다. 그러나 WYSIWYG에 익숙한 사람에게 직관적으로 사용할 만해 보이지 않는다는 문제가 있다.

간단히 말해, 워드 프로세서 쓸 때는 이러저러하게 했던 그런 기능을 LATEX으로 해보고

[1] 옛날에 나는 나의 학위논문을 128k 램을 가진 Z80 가정용 컴퓨터에서 작성했다. WordStar 워드 프로세서와 나의 논문이 단면 CP/M 부팅가능 720k 플로피 디스크 한 장에 다 들어가고도 남는 공간이 많았다.

싶을 때가 있다. 한때 좋아했던 :-) 워드 프로세서로 했던 것과 같은 결과를 얻으려면 LaTeX으로 어떻게 하면 되는지 알 수 있다면 좋겠다는 것이다.

이것이 내가 이 짧은 안내서를 쓰게 된 동기이다. 이미 말한대로 나는 LaTeX의 기본을 알고 있는 사람을 전제로 하고 있다. 만약 당신이 진정한 초심자라면 http://www.ctan.org/starter.html로 가서 'The (Not So) Short Introduction to LaTeX 2_ε'를 읽어볼 것을 권한다. 그 외의 좋은 처음 시작 문서로 http://en.wikibooks.org/wiki/LaTeX/이 있다.

한국어 번역본을 읽으려면 명령행에서 `texdoc lshort-kr`.

이어지는 절에서 어떤 가상의 워드 프로세서를 상정하고 그 메뉴와 메뉴 항목을 살펴보면서 각각에 대응하는 LaTeX 방식을 찾아보겠다.

1.1 기본사항

워드 프로세서의 기능 가운데 많은 부분은 에디터의 몫이다. 그밖에 LaTeX 명령에 의해 구현되는 것이 있고 패키지(*packages*)에 의해 이루어지는 것이 있다. 패키지란 LaTeX을 확장하여 새로운 명령과 환경을 제공하는 매크로 모음이다. 수많은 패키지가 존재하는데 유일한 문제는 그게 어디에 있는지, 무슨 일을 하는지, 그리고 사용하려면 어떤 절차를 거쳐야 하는지를 알아야 한다는 것이다. 패키지에 대해서는 1.1절에서 더 다룬다.

`\usepackage` 명령으로 쓸 수 있는 것을 보통 '패키지'라 한다. 이것은 스타일 파일(`.sty`)과 설정 파일 및 사용법 문서로 이루어져 있다.

패키지를 포함하여 그밖의 TeX에 관련된 자료들은 CTAN (the Comprehensive TeX Archive Network)을 구성하는 여러 사이트를 통해 이용할 수 있다. http://www.ctan.org 사이트를 이미 소개했다. 이 사이트는 여러 곳의 미러 사이트를 가지고 있다. 이제부터 CTAN:이라 하면 "자신이 선호하는 CTAN 미러 사이트의 TeX 디렉터리"를 가리키는 것으로 하겠다. 예를 들면 자신의 플랫폼에 적합한 LaTeX을 얻기 위해서는 CTAN://systems (여기서는 http://www.tex.ac.uk/tex-archive/systems/)에 접속하면 된다.

KTUG에서도 CTAN을 미러링 한다.

문서 작성을 위해서는 좋은 텍스트 에디터가 필요하다. 초심자에게는 *LaTeX shell*, 즉 소스를 작성하고 미리보기를 할 수 있는 등 LaTeX에 필요한 기능을 갖춘 *LaTeX* 전용 에디터가 더 좋을 것이다.

아래에 나열된 프로그램은 추천할 만한 것이다. 이 모두가 Free/Open Source 소프트웨어들이다.

- Texmaker (multiplatform):
  http://www.xm1math.net/texmaker/index.html
- TeXstudio (multiplatform):
  http://texstudio.sourceforge.net/
- TeXworks (multiplatform):
  http://tug.org/texworks/
- TeXShop (Mac OS X):
  http://www.uoregon.edu/~koch/texshop/
- TeXnicCenter (Windows):
  http://www.texniccenter.org/

매킨토시에서 LaTeX을 활용하는 문제에 관한 정보를 얻으려면 http://www.esm.psu.edu/mac-tex/을 참고하라.

에디터가 지원하는 기능

LaTeX은 조판기일 뿐이다. 잘라붙이기, 찾기와 바꾸기 등은 에디터에게 맡겨진다. 표 1은 geek들에게 유명한 편집기인 GNU emacs와 vim의 기본 키 바인딩과 Borland IDE 키 바인딩을 적용한 jed의 주요 명령을 요약한 것이다.

Action	Emacs	Vim	Jed
command mode	Alt-X	ESC	Alt-X
insert mode	n/a	i a o O	n/a
line editor mode	n/a	:	n/a
	file operations		
open file	Ctrl-X Ctrl-F	:e	Ctrl-KE
insert file	Ctrl-Xi	:r	Ctrl-KR
save file	Ctrl-X Ctrl-S	:w	Ctrl-KD
save as	Ctrl-X Ctrl-W name	:w name	Ctrl-KS
close file	Ctrl-XK	:q	Ctrl-KQ
change buffer	Ctrl-XB	bN	Ctrl-KN
undo	Ctrl-XU	u	Ctrl-U
redo	Ctrl^	Ctrl-R	Ctrl-G Ctrl-U
exit	Ctrl-X Ctrl-C	:qa!	Ctrl-KX
	moving around		
word left	Alt-B	b	Ctrl-A
word right	Alt-F	w	Ctrl-F
start of line	Ctrl-A	0	Ctrl-QS
end of line	Ctrl-E	$	Ctrl-QD
page up	Alt-V	Ctrl-U	Ctrl-R
page down	Ctrl-V	Ctrl-D	Ctrl-C
start of buffer	Alt-<	1G	Ctrl-QR
end of buffer	Alt->	G	Ctrl-QC
line n.	Alt-G n.	n.G	Ctrl-QI
	deleting		
character left	Ctrl-H	X	BS
character right	Ctrl-D	x	Alt-G
word left	Alt-DEL	db	Alt-BS
word right	Alt-D	dw	Ctrl-T
end of line	Ctrl-K	d$	Ctrl-QY
line	Ctrl-A Ctrl-K	dd	Ctrl-Y
	search & replace		
search	Ctrl-S text	/text	Ctrl-QS
replace	Alt-%	:s/old/new/g	Ctrl-QA
	blocks		
start selection	Ctrl-SPACE	v	Ctrl-KB
cut	Ctrl-W	D	Ctrl-KY
copy	Alt-W	Y	Ctrl-KH
paste	Ctrl-Y	P	Ctrl-KC

표 1: Emacs, Vim, Jed의 유용한 단축키

패키지 추가

아래의 사항은 TeX Live에 적용된다. TeX Live는 대부분의 GNU/Linux 배포판에 포함되어 있다. MacTeX에도 적용될 것이지만 내가 직접 경험해보지는 못했다. MiKTEX(아마 가장 유명한 Windows 텍 시스템)을 위한 안내는 그 뒤에 이어진다.

대부분의 필요한 패키지는 TeX Live 자체가 지원하는 것으로 충분하다.

방대한 분량의 LATEX 패키지들이 기본으로 지원된다. 예를 들면 Ubuntu는 많은 texlive-* 패키지를 제공한다. 이것은 우분투의 .deb 패키지를 가리키는 것이므로 tex live 자체와 혼동하지 말하야 한다.

만약 지원되지 않는 패키지를 사용해야 한다면 아래와 같이 하라.

Windows용 TeX Live의 경우는 %USERPROFILE%\texmf 아래에 본문의 설명과 같은 디렉터리 구조를 만든다. 단, Windows라도 %HOME% 변수가 설정되어 있다면 그 아래로 가야 한다.

1. 다음과 같은 디렉터리 구조를 만든다.

   ```
   $ mkdir -p ~/texmf/tex/latex
   ```

   이 디렉터리 아래 새로운 패키지를 가져다 둘 것이다.

2. 가까운 CTAN 미러 사이트에서 패키지를 (대부분 zip-압축 디렉터리이다) 내려받는다. 예컨대 foo.zip이라 하자.

3. 적당한 곳에 압축을 푼다.

   ```
   $ mkdir ~/texmf/tex/latex/foo
   $ mv foo.zip ~/texmf/tex/latex/foo
   $ cd ~/texmf/tex/latex/foo ; unzip foo.zip
   ```

4. latex foo.ins or latex foo.dtx to create it; 만약 .sty 파일이 없으면 latex foo.ins 또는 latex foo.dtx 를 실행하여 만들어내게 한다.

5. texhash ~/texmf 명령을 실행한다.

최근의 tex live 배포판에서는 home/texmf 아래 설치한 파일들에 대하여 texhash나 mktexlsr를 실행하지 않아도 되게 되어 있다.

MiKTEX에 새로운 패키지를 추가하려면 \latex\newpackage를 C:\localtexmf\tex\ 아래 만들고 거기에 필요한 파일을 가져다 둔다. 앞서와 같은 과정을 진행한 다음 MiKTeX Options를 실행하여 Refresh now 버튼을 클릭한다. 또는 initexmf -u 명령을 실행한다. 그것으로 되었다.

MiKTeX에 관한 이 지침은 MiKTeX 2.4 이전 버전에 해당하는 것 같다. 현재 버전에서는 위치가 조금 다르다. 사용자 패키지를 MiKTeX 2.8 또는 2.9에서 설치하기 위해서는 두 가지 방법이 있는데 하나는 %APPDATA% 또는 %PROGRAMDATA% 아래 설치된 사용자용 texmf 트리를 이용하는 것이고 다른 하나는 이전과 비슷하게 localtexmf를 추가하는 것이다. 다만 두번째 경우 반드시 사용자가 TeX roots를 추가하는 조작을 해주어야 한다.

일단 패키지가 설치되면 \documentclass 선언 이후에 다음 한 줄 추가하는 것으로 자신의 문서에 사용하게 할 수 있다.

```
\usepackage{foo}
```

Info 페이지 추가

'man'과 'info' 페이지는 UNIX와 리눅스 운영체제에서 일반적으로 채용하고 있는 명령행 소프트웨어 도움말 문서이다. 만약 자신의 LATEX 배포판이 latex2e.info라는 도움말 파일을 설치해주지 않는다면, 다음과 같이 하라.

1. http://tug.ctan.org/info/latex2e-help-texinfo/latex2e.info에서 다운로드받는다.

2. 다음 명령을 실행한다.

```
$ sudo cp latex2e.info /usr/share/info/
$ sudo ginstall-info latex2e.info dir
```

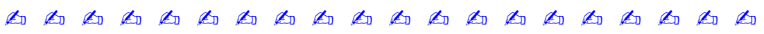
이제 `info latex` 또는 `info latex2e` 명령으로 latex2e의 info 페이지를 볼 수 있다.

1.2 황금률

계속하기 전에, 명심해야 할 것이 있다.

1. 문서의 **구조화**에 익숙해져야 한다. 편, 장, 절과 같은 체제를 염두에 두도록 하라. 딱히 과학분야 문서를 쓰는 것이 아니어도 이것을 지키는 것이 좋다.
2. LaTeX은 확실히 포매팅 파라미터로 문서를 난잡하게 만드는 것을 기피하게 한다. **모양**에 너무 신경쓰지 말고 **내용**에 집중하라.

이 간단한 규칙을 지키면 실제 인쇄된 출력물은 신기하게도 멋진 결과로 나올 것이다.
그렇지만, 이 안내서는 두 번째 규칙을 극복하는 방법을 (조금 어기는 한이 있더라도) 알려준다. 비구조적인 문서, 이를테면 회람문, 공지서, 포스터 등등을 작성할 수도 있게 해줄 것이다.



2 File 파일 메뉴

이 메뉴에 속하는 몇몇 항목은 LaTeX과 별 상관없음이 명백하다. File/Open, File/Save, File/Close 등은 에디터가 할 일이다.

2.1 File/New 새 파일

빈 문서에 해당하는 LaTeX 소스는 다음과 같다.

```
\documentclass{article}
\thispagestyle{empty} % no page number
\begin{document}
% This is a comment. Write your stuff here.
\end{document}
```

LaTeX으로 쓰여진 문서는 본질적으로 구조화되어 있다. 좀더 구체적인 보기는 다음과 같다.

```
\documentclass[a4paper,12pt]{article}
\begin{document}
\title{My Document}
\author{John Smith}
\date{London, \today}
\maketitle
\begin{abstract}
This is a very short article.
\end{abstract}
```

```
\tableofcontents
\listoftables
\listoffigures
\section{First Section}
\label{sec:start}
This is the text of the section. See \cite{Gonzato} for details.
\section{End}
\label{sec:end}
This is the end of the document. Please go to Section
\ref{sec:start} to read it again.
\begin{thebibliography}{99}
\bibitem{Gonzato} Gonzato G. \textit{\LaTeX{} for Word Processor
Users}. CTAN, 2001--2015.
\end{thebibliography}
\end{document}
```

더 많은 문서 표본이 부록 A에 있다.

2.2 File/Save As… 다른 이름으로 저장

다음 도구들은 LaTeX을 다른 포맷으로 변환할 때 유용하다.

- TeX4ht는 LaTeX을 HTML로 변환하는 가장 훌륭한 변환기일 것이다.
  http://tug.org/tex4ht
- HTML 변환기로 latex2html이라는 것이 있다.
  http://saftsack.fs.uni-bayreuth.de/~latex2ht/,
  CTAN://support/latex2html
- latex2rtf는 Rich Text Format으로 변환한다.
  CTAN://support/latex2rtf
- detex은 명령행 프로그램으로서 LaTeX의 모든 태그(명령)를 제거하고 plain text로 바꾼다.
  http://www.cs.purdue.edu/homes/trinkle/detex/,
  CTAN://support/detex/

PDF 파일로 변환하는 문제에 대해서는 2.7절에서 자세히 다룬다.

2.3 File/Save As Template 본보기 문서로 저장

LaTeX 템플리트를 저장한다는 말의 의미가 만약 LaTeX 패키지를 작성하는 것을 의미한다면, 그것은 너무 복잡한 문제라서 이 안내서가 다룰 수 있는 범위를 넘어선다.

2.4 File/Import 다른 형식의 문서 들여오기

다음 도구들은 다른 포맷으로부터 LaTeX으로 변환한다.

sourceforge에 rtf2latex2e라는 것이 있는데 사용 언어에 제한이 있고 한글은 포함되어 있지 않다.

- rtf2latex: CTAN://support/rtf2latex
- html2latex: CTAN://support/html2latex

- `wvware`는 MS Word를 LaTeX을 포함하여 여러 포맷으로 변환하는 도구의 모음이다. http://wvware.sourceforge.net
- Abiword는 프리 워드 프로세서이다. http://www.abiword.org, MS Word 문서를 불러올 수 있고 LaTeX으로 저장 가능하다.
- `txt2tex`: CTAN://support/txt2tex 플레인 텍스트 파일을 LaTeX으로 변환하는데 꽤 좋은 결과를 보여준다.

그밖의 `*2latex` 컨버터를 같은 주소에서 찾아볼 수 있다.

또 한 가지 재미있는 것은 OOoLatex이라는 OpenOffice 확장 매크로이다. http://ooolatex.sourceforge.net 리브레오피스 사용자는 TexMaths라는 확장 플러그인을 쓸 수 있다. http://roland65.free.fr/texmaths/.

2.5 File/Page Setup 쪽 설정

페이지 크기, 방향, 여백을 설정하는 일반적인 방법은 `\documentclass`의 인자로 이를 지정하는 것이다. 페이지 크기는 `a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper`, `executivepaper` 중에서 고를 수 있고, 방향은 `portrait`가 디폴트이며 `landscape`를 사용할 수 있다. 예를 들어보자.

```
\documentclass[a5paper,landscape,12pt]{article}
```

문서 전체에 걸친 여백(margin)은 페이지 레이아웃 길이값 변수를 `\setlength` 명령으로 변경한다.

이와 같이 레이아웃 길이값 변수를 직접 바꾸는 것은 불편할 뿐 아니라 더 복잡한 클래스에서 의도하지 않은 충돌이나 불충분한 설정이 발생할 수 있다. 아래에서 소개하는 geometry 패키지를 사용하여라.

```
\setlength{\leftmargin}{2cm}
\setlength{\rightmargin}{2cm}
\setlength{\oddsidemargin}{2cm}
\setlength{\evensidemargin}{2cm}
\setlength{\topmargin}{-1cm}
\setlength{\textwidth}{18cm}
\setlength{\textheight}{25cm}
```

geometry 패키지는 페이지 크기, 여백 너비 등등의 파라미터를 완벽하게 제어할 수 있게 해준다. 이 패키지의 옵션은 너무 많아서 그 사용법을 모두 열거하는 것이 어려우므로 문서를 꼭 읽어볼 것을 권한다. 다음에 보인 몇 가지 예들을 참고할 수 있을 것이다. 여기서 일부 설정은 다른 설정과 충돌할 수 있다. 예시로서 제시한 것임에 유념하라.

memoir에서는 geometry를 이용하기 보다 memoir 자체의 레이아웃 설정 기능을 이용하는 것이 좋다. oblivoir에는 fapapersize라는 간편한 판면 설정 패키지가 제공된다.

```
\usepackage{geometry}  % top of document
...
\geometry{paperwidth=25cm}
\geometry{paperheight=35cm}
% or: \geometry{papersize={25cm,35cm}}
\geometry{width=20cm}  % total width
\geometry{heigth=30cm} % total heigth
% or: \geometry{total={20cm,30cm}}
\geometry{textwidth=18cm}  % width - marginpar
\geometry{textheight=25cm} % heigth - header - footer
% or: \geometry{body={18cm,25cm}}
```

```
\geometry{left=3cm}    % left margin
\geometry{right=1.5cm} % right margin
% or: \geometry{hmargin={3cm,2cm}}
\geometry{top=2cm}     % top margin
\geometry{bottom=3cm}  % bottom margin
% or: \geometry{vmargin={2cm,3cm}}
\geometry{marginparwidth=2cm}
\geometry{head=1cm}    % header space
```

각 옵션은 다음처럼 줄 수 있다.

```
\usepackage[left=3cm, right=2cm]{geometry}
```

Page Setup/Headers and Footers 머릿글과 바닥글

memoir 또는 oblivoir 클래스는 fancyhdr와는 다른 방식으로 페이지 스타일을 설정한다. 훨씬 직관적이고 강력한 커스터마이징이 가능하다. 자세한 것은 memoir 사용설명서를 참고하라. 여기에 주석을 붙이는 이유는 fancyhdr와 일부 상충하는 경우가 있으므로 이를 미리 알려두고자 함이다. 즉, memoir에서는 fancyhdr를 쓰지 않는 것이 좋다.

fancyhdr 패키지는 `\pagestyle{fancy}`라는 새로운 명령을 제공한다. 이것은 현재 섹션(또는 `book.cls`에서는 챕터)과 서브섹션으로 헤더를 만들고 페이지 번호를 바닥에 찍어준다. 제법 팬시하다. 머리말과 꼬리말은 당연히 사용자가 변경할 수 있다. 세 부분으로 이루어지는데, 각각 왼쪽으로 정렬되는 부분, 가운데 오는 부분, 오른쪽으로 정렬되는 부분이다. 이들을 사용자가 설정하려면 다음 보기와 같이 한다.

```
\usepackage{fancyhdr}
...
\lhead{} % empty
\chead{Hello, world!}
\rhead{Page \thepage} % page number
\lfoot{}
\cfoot{\textbf{Hello!}}
\rfoot{}
```

2.6 File/Printer Setup 프린터 설정

이것은 운영체제 의존적인 문제로서 LaTeX과는 아무 상관없는 것이다. 만약 Unix 계열의 시스템을 사용한다면 다음 팁이 도움이 될 것이다.

- `lpr -P printername` 지정된 프린터로 출력.
- `lpr -# 10` 10부 출력
- `lpr -r` 프린트 후 파일 삭제.

2.7 File/Print Preview 인쇄 미리보기

LaTeX 입력 파일이 준비되었다면 다음 중에서 선택할 수 있다.

- `.dvi`로 변환(`latex file.tex`)하여 `xdvi`나 `yap`과 같은 프리뷰어로 미리보기할 수 있다.
- `.dvi`를 `dvips`를 통하여 PostScript로 변환한다. 그런 다음에 `Ghostview`와 같은 프로그램으로 미리보기한다.

- dvipdf를 이용하여 .dvi를 .pdf로 변환하거나 직접 pdflatex으로 .pdf 파일을 생성한다.

오늘날 .dvi나 .ps 파일 출력은 거의 문제삼지 않게 되어 간다. 예를 들어 차세대 TEX 엔진 LuaTEX이나 XƎTEX도 .pdf 출력이 기본이다.

내 생각에, .pdf 파일을 만드는 것이 가장 좋다. 활용가능성이 가장 폭넓기 때문이다.

dvipdf는 .dvi를 .ps를 거쳐 .pdf로 만드는 스크립트일 뿐이다. 실제로는, pdflatex을 사용하는 것은 더 흥미롭다. 사실 hyperref나 url과 같은 패키지들은 .pdf 파일이 브로우징 가능하게 만들어준다. 4.15절을 보라. 그러나 pdflatex을 사용하려면 주의해야 할 점이 좀 있다. 다른 패키지와의 호환성 문제를 경험할 수도 있기 때문이다. 자세한 것은 4.12절을 보라.

2.8 File/Print 인쇄

(UNIX 계열 운영체제에서) 간단히 lpr file.ps를 명령행에서 주거나 프리뷰어의 File/Print 메뉴 항목을 선택하면 된다.

2.9 File/Versions 버전관리

version 패키지가 LATEX 소스의 버전 관리를 위한 초보적인 기능을 제공하기는 하지만 외부 유틸리티를 사용하는 것이 더 나은 선택이다.

리비전 컨트롤과 협업을 위한 프로그램은 무척 많다. 예전 스타일 사용자들은 RCS같은 단일 사용자 도구를 쓰는 경우고 있고 (저자가 그러하다), Subversion, Git, Mercurial과 같은 강력한 다중 사용자 도구를 선호하는 사람도 많다. 이 기능을 내장한 에디터도 있다.

LATEX과 Subversion에 대한 소개글을 다음에서 찾아볼 수 있다.
http://tug.org/pracjourn/2007-3/kalderon-svnmulti/.

3 Edit 편집 메뉴

이 메뉴는 LATEX 기능보다는 에디터와 더 많이 관계된다. 편집/잘라내기, 편집/복사하기, 편집/붙이기, 편집/찾기, 편집/바꾸기와 같은 에디터에 공통되는 항목들에 대한 단축키는 표 1에서 이미 보였다.

텍스트 일부를 선택하는 것은 자르기, 붙이기를 위해서이기도 하지만 선택된 텍스트에 특정 스타일을 적용하기 위해서이기도 하다. 이에 대응하는 LATEX의 작용은 텍스트 일부를 중괄호나 환경 (environment)으로 감싸는 것이다. 예를 들면 텍스트 일부에 두꺼운 글씨 속성을 부과하려면 다음 가운데 한 가지 방식을 쓰면 된다.

```
1  this is \textbf{bold text;}\\
2  this is also
3  {\bfseries bold text;}\\
4  \begin{bfseries}
5  this is bold text, too!
6  \end{bfseries}
```

this is **bold text;**
this is also **bold text;**
this is bold text, too!

3.1 Edit/Autotext 자동완성

자동 완성이란 예를 들어 'PS'라고 입력하면 'PostScript'라고 자동으로 입력되는 기능을 말한다. 이것도 에디터의 역할이지만 대략 여기에 해당하는 LaTeX 기능이 있다.

```
\def\PS    {\textsc{PostScript}}
```

이렇게 하면 \PS라고 입력하는 곳마다 \textsc{PostScript}에 해당하는 PostScript가 찍힌다. 대소문자 구별에 주의하자.

사용자 수준에서는 일반적으로 \def보다 \newcommand를 쓰는 것이 더 안전하다.

4 Insert 삽입 메뉴

4.1 Insert/Breaks 행 나누기, 쪽 나누기

- 행 나누기가 일어나지 않는 공백은 ~ (틸데) 문자로 표시한다.
- 강제 행 나누기를 하려면 \linebreak나 \newline을 쓴다. 두 명령의 차이는 아래 예제를 보라. \\ 명령도 새로운 줄을 시작한다. 이 명령에는 \\[1cm]과 같이 옵션 인자로 행 간격을 줄 수 있다.
- 새 문단을 시작하려면 빈 줄을 두거나 \par 명령을 쓴다.
- 끝으로 강제 페이지 나누기는 \newpage나 \clearpage를 쓴다.

원문에는 \\를 새 문단으로 소개하고 있는데, 이것은 \par와 같은 것이 아니라 \newline과 같은 것이다. 설명하는 순서를 조금 바꾸었다.

\linebreak와 \newline의 차이는 앞의 것이 행의 나머지를 다 채우고 다음 행을 시작한다는 것이다.

또한, \clearpage는 \newpage와 마찬가지로 새 페이지를 시작하지만 그 시점까지 출력되지 않고 대기중인 *floats*들, 즉 figure나 table들을 모두 출력한 다음에 새 페이지를 만든다는 점이 다르다. float에 대해서는 4.12절에서 설명한다.

다음 보기는 \linebreak와 \newline의 차이를 보인 것이다.

```
1  I am stretched!\linebreak
2  But I am not.\newline
3  Ok, now you get it.
```

I am stretched!
But I am not.
Ok, now you get it.

4.2 Insert/Enumerated List 리스트 문단

숫자나 기호 붙은 리스트는 itemize와 enumerate 환경에 해당한다. 리스트 환경에서 글머리에 붙는 기호를 바꾸려면 \item 명령의 옵션 인자로 이를 특정하면 된다.

```
1  \begin{itemize}
2    \item[*] with an asterisk;
3    \item[-] with a dash;
4    \item[.] with a dot.
5  \end{itemize}
```

* with an asterisk;
- with a dash;
. with a dot.

다른 방법은 **카운터**(*counters*) 수식 스타일을 재정의하는 것이다.[2] 리스트 문단의 제1수준에서 제4수준까지의 번호기 매크로는, itemize에서 `\labelitemi`, `\labelitemii`, `\labelitemiii`, `\labelitemiv`이고 enumerate에서 `\labelenumi`, `\labelenumii`, `\labelenumiii`, `\labelenumiv`이다.

다음 보기에서 `\labelitemi`와 `\labelitemii`를 재정의하여 `itemize`의 문단 머리 번호기 모양을 바꾼 예를 보였다.

```
1  \begin{itemize}
2  \renewcommand{\labelitemi}{*}
3  \renewcommand{\labelitemii}{-}
4    \item first level, item 1
5    \item first level, item 2
6    \begin{itemize}
7      \item second level, item 1
8      \item second level, item 2
9    \end{itemize}
10   \item first level, item 3
11 \end{itemize}
```

* first level, item 1
* first level, item 2
    - second level, item 1
    - second level, item 2
* first level, item 3

카운터 수식자는 `\arabic`이 보통 숫자, `\roman`은 소문자 로마 숫자(예: viii, ix), `\Roman`이 대문자 로마 숫자, `\alph`와 `\Alph`가 각각 소문자와 대문자 알파벳, 그리고 나중에 설명할 `\fnsymbol`이 있다.

그러므로 enumerate 리스트에서 알파벳 대문자와 소문자 로마 숫자를 사용하려면 다음 보기와 같이 하면 된다.

```
1  \begin{enumerate}
2  \renewcommand{\labelenumi}
3    {\Alph{enumi}}
4  \renewcommand{\labelenumii}
5    {\roman{enumii}}
6    \item first level, item 1
7    \item first level, item 2
8    \begin{enumerate}
9      \item second level, item 1
10     \item second level, item 2
11   \end{enumerate}
12   \item first level, item 3
13 \end{enumerate}
```

A first level, item 1

B first level, item 2

i second level, item 1

ii second level, item 2

C first level, item 3

또다른 방법으로 enumerate 패키지를 사용할 수 있다. 이 패키지는 `enumerate` 환경을 재정의하여 옵션 인자로 주어진 형식을 쓰게 한다. `A a I i 1` 가운데 한 글자가 오면 그것을 카운터 값에 각각 `\Alph`, `\alph`, `\Roman`, `\roman`, `\arabic`이 주어진 것으로 대치한다. 그밖의 글자로 이루어진 텍스트를 글머리에 쓰려면 중괄호로 해당 텍스트를

[2] LaTeX의 숫자 표현(절, 리스트, 그림 등)에는 자동할당되는 카운터가 있다.

묶어주면 된다.

다음 예는 enumerate 패키지의 방법으로 enumerate의 문단 머리 번호기를 바꾸는 방법을 보인 것이다.

```
\begin{enumerate}[{Example} I.]
  \item First example.\label{item:first}
  \item Second example.
  \item Last example.
    Go to Item~\ref{item:first}.
\end{enumerate}
```

Example I. First example.

Example II. Second example.

Example III. Last example. Go to Item I.

[역자 보충] 참고로, memoir에는 enumerate 패키지가 이미 포함되어 있어서 별다른 조치 없이 바로 위 문단에서 설명한 번호기 옵션 인자를 바로 쓸 수 있다. 이 문서에서는 언급하고 있지 않지만 enumitem 패키지도 이와 유사한 기능을 제공한다. koTEX에는 enumerate에 해당하는 dhucs-enumerate, paralist에 해당하는 dhucs-paralist 패키지를 제공하고 있으며 이 패키지들은 한글식의 문단머리를 지정할 수 있게 하고 있다. 추가된 글머리표지는 가, ①, ⑴, ⓐ, ⓐ, ⅰ, Ⅰ, ㄱ, ㉠, ㉮, ㉰, ㉮㉮의 열두 개이다. 다음 한글 사용 사례는 dhucs-enumerate 패키지가 필요하다.

```
\begin{enumerate}[{보기} ㉮]
  \item 번호붙은 문단\label{item:a}
  \item 샘플
  \item 아이템~\ref{item:a}\를 참조.
\end{enumerate}
```

보기 ㉮ 번호붙은 문단

보기 ㉯ 샘플

보기 ㉰ 아이템 ㉮를 참조.

enumerate에서 번호를 임의로 붙이려면 카운터를 재정의한다.

```
\begin{enumerate}
  \setcounter{enumi}{2}
  \item Example 3.
  \item Example 4.
  \setcounter{enumi}{5}
  \item Example 6.
\end{enumerate}
```

3. Example 3.

4. Example 4.

6. Example 6.

리스트는 item 사이에 간격을 주는 독립 문단으로 식자된다. 반면, 열거 리스트를 사용하지만 이것을 각각의 문단으로 식자하지 않고 한 문단 내에서 나열 항목만을 나타내고자 할 때가 있다. paralist 패키지를 이용한다. 이 패키지도 enumerate와 마찬가지로 번호기를 선택 인자로 지정할 수 있다. `inparaenum` 환경을 쓸 수 있다.

다음 예제에서 보는 대로 enumerate에서와 마찬가지로 `A a I i 1` 문자는 카운터를 어떤 수식자로 표현할지를 나타낸다. paralist는 더 많은 일을 할 수 있다. 해당 패키지 문서를 읽어볼 것을 권장한다.

```
I'll throw in a list of items:
\begin{inparaenum}
  \item apples,  \item pears, and
  \item oranges.
\end{inparaenum}
The same list can be labelled
with letters:
\begin{inparaenum}
  [\itshape a) \upshape]
  \item apples, \label{first}
  \item pears, and
  \item oranges. The first item is \ref{first}.
\end{inparaenum}
```

I'll throw in a list of items: 1. apples, 2. pears, and 3. oranges. The same list can be labelled with letters: *a)* apples, *b)* pears, and *c)* oranges. The first item is a.

[역자] 다음 한글 사용례는 xob-paralist를 이용한 것이다.

```
문단 안에서 항목을 나열하고자 한다.
\begin{inparaenum}[ㄱ)]
  \item 사과, \label{firstk}
  \item 배, 그리고  \item 복숭아.
\end{inparaenum}
다른 방식의 라벨을 달아본다.
\begin{inparaenum}[①]
  \item 사과,  \item 배, 그리고
  \item 복숭아.
  첫번째 아이템을 참조하면, \ref{firstk}.
\end{inparaenum}
```

문단 안에서 항목을 나열하고자 한다. ㄱ) 사과, ㄴ) 배, 그리고 ㄷ) 복숭아. 다른 방식의 라벨을 달아본다. ① 사과, ② 배, 그리고 ③ 복숭아. 첫번째 아이템을 참조하면, ㄱ.

4.3 Insert/Special Character 특수문자

먼저 LaTeX 입력 파일을 만들 때 몇 가지 글자는 입력 방법이 정해져 있다는 것을 상기하자. 예를 들면 여는 따옴표는 ``로, 닫는 따옴표는 ''로 입력하는 것이 관행이다. 그리고 --는 en-dash(–), ---는 em-dash(—)를 입력하는 방법이다. '{'와 '}'는 특히 중요한데 이것은 LaTeX에서 '범위'를 설정하는 기호로 쓰이기 때문에 이대로 입력하여도 텍스트에 나타나지 않는다. 이 괄호 기호를 텍스트로 나타내고 싶다면 반드시 \{와 같이 입력해야 한다. LaTeX의 문법과 관련된 특수기호(이른바 '예약문자')는 { } ^ _ # $ % ~ \ &와 같은 것이 있다. 이 문자를 그대로 입력하면 오류를 만날 수 있다.

이 한 문단은 역자가 상세히 설명하기 위해 부연하였다.

이와 같이, 몇 가지 문자들은 LaTeX에서 특별한 의미를 갖는다는 것을 명심해야 한다. 이 글자들을 문장에 나오게 하려면 \를 앞에 붙이거나 수학 모드에서 쓰거나 하는 등 특별한 방법으로 입력하여야 한다. 표 2를 보라. 이 표에 있는 일부 명령은 textcomp 패키지가 필요하다.

textcomp 패키지에 관한 언급은 역자의 추가임.

유럽어의 액선트 붙은 문자를 예를 들어 \'e와 같이 입력하여 'é'를 얻는 것은 비슷하지만 이와는 또다른 문제이다. 이에 대해서는 7.3절을 보라.

특수 문자를 입력하는 또 한 가지 방법은 해당 ASCII 코드를 \char 명령으로 주는 것이다.

ASCII 코드가 아니라 유니코드를 사용하는 XeTeX과 같은 엔진에서는 이 방법이 뜻대로 되지 않을 수 있다. 대체로 \char를 이용하는 방법은 되도록 쓰지 않는 것이 좋다.

Character	LaTeX Sequence
$	\$ or \textdollar
&	\& or \textampersand
%	\% or \textpercent
_	_ or \textunderscore
{	\{ or \textbraceleft
}	\} or \textbraceright
<	$<$ or \textless
>	$>$ or \textgreater
\	\textbackslash
#	\#
|	\textbar
•	\textbullet
‡	\textdaggerdbl
†	\textdagger
¶	\textparagraph
§	\textsection
©	\textcopyright
^	\textasciicircum
~	\textasciitilde or \~{}
∼	$\sim$
®	\textregistered
™	\texttrademark
ª	\textordfeminine
º	\textordmasculine

표 2: 특수문자 입력 방법

예를 들면 $ & ^ ~를 얻으려면 \char36 \char38 \char94 \char126이라고 입력한다.

특별한 글자와 심볼을 많이 제공하는 패키지가 있다. 예를 들면 pifont는 \ding, \dingfill, \dingline, \dinglist와 같은 명령을 제공한다. 첫번째의 \ding 명령은 특정 코드에 해당하는 딩뱃 문자를 찍어준다. 다른 명령들은 각각 특정 딩뱃 코드를 인자로 줄 수 있는 \fill, \line, \list들이다. 예를 들면 \dinglist 환경을 다음처럼 쓰면 딩뱃 문자를 번호기로 하는 \list를 만들 수 있다.

```
\begin{dinglist}{43}
  \item one
  \item two
  \item three
\end{dinglist}
```

☞ one

☞ two

☞ three

다음 보기는 좀더 그럴싸하다.

```
\begin{dingautolist}{172}
  \item one
  \item two
  \item three
\end{dingautolist}
```

① one

② two

③ three

심볼 문자는 여기서 언급하기에 너무 그 수가 많아서 차라리 'The Comprehensive LaTeX Symbol List'라는 문서를 읽어보는 편이 낫다. CTAN://info/symbols/comprehensive에서 찾을 수 있다.

대부분의 TeX 배포판에서 texdoc을 이용하여 texdoc symbols-a4라고 명령행에서 치면 바로 읽을 수 있다.

유로화 기호 (€)

공식 유로화 기호는 eurosym 패키지가 제공한다. 이것은 다음 두 가지 방법으로 사용 가능하다.

XeTeX에서는 굳이 eurosym이나 marvosym을 쓸 것 없이 바로 유로화 기호를 €([U+20AC]) 입력하거나 `\char"20AC`와 같이 입력할 수도 있다. 일반적으로 `\texteuro`(€)를 쓰는 것이 좋다. 요즘 대부분의 폰트에는 유로화 기호가 들어 있다. 참고로, `\textwon` ₩과 `\textyen` ¥도 있다.

```
\usepackage[gen]{eurosym}
\usepackage[official]{eurosym}
```

둘 다 `\euro` 명령을 제공하며 결과는 €로 나타난다. 이 기호가 실제 찍히는 모양은 `[gen]` 옵션을 주느냐 `[official]` 옵션을 주느냐에 달려 있는데, `[gen]`의 경우는 €로 찍히고 `[official]`의 경우는 €로 찍힌다. 차이점을 눈여겨 보라. 두 번째 것은 `\officialeuro` 명령으로도 얻을 수 있다.

marvosym 패키지도 유로 기호를 제공한다. 이 패키지는 이외에도 꽤 많은 멋진 기호 문자를 포함하고 있다. 이 패키지를 쓸 때 유로화 기호는 `\EUR` 명령으로 €과 같은 결과를 얻는다.

4.4 Insert/Formula 수학식

LaTeX은 특히 수식 조판에 강하다. 수학 기호를 텍스트 속에 넣으려면 그것을 `$`로 감싸주어야 한다. LaTeX에서는 문장 중 수식을 `\(`와 `\)`로 감싸주는데 이것이 `$`로 시작과 끝을 동시에 표시하는 것보다 입력 실수를 줄여준다.

역자가 보충

```
1  I like math: $x^n + y^n \neq
2  z^n, \forall n \neq 2$
3  is my favourite theorem.
```

I like math: $x^n + y^n \neq z^n, \forall n \neq 2$ is my favourite theorem.

`displaymath`와 `equation`은 별행 수식을 식자한다. `equation`은 교차참조할 수 있도록 수식 번호를 붙여준다. `\begin{displaymath}`와 `\end{displaymath}`는 `\[`와 `\]`로 대신 쓸 수 있다.

`displaymath`나 `equation`은 그 안에서 줄나누기가 되지 않는다. `eqnarray`는 되도록 쓰지 않는 것이 좋다. 여러 줄 수식은 $\mathcal{AMS}$-math를 이용하여라. 수식 입력에 관한 사항은 별도의 참고문헌을 보아야 한다. 특히 mathmode라는 문서가 유명하다.

```
1  Fermat's Last Theorem is
2  defined as:
3  \begin{equation}
4  x^n + y^n \neq z^n,
5  \forall n \neq 2
6  \label{eq:fermat}
7  \end{equation}
8  Can you prove
9  Eq.~\ref{eq:fermat}?
```

Fermat's Last Theorem is defined as:

$$x^n + y^n \neq z^n, \forall n \neq 2 \tag{1}$$

Can you prove Eq. 1?

4.5 Insert/Footnote 각주

`\footnote[n]{footnotetext}` 명령으로 충분하다. 선택적 인자인 `[n]`은 각주 번호를 임의로 설정한다. `\footnote` 명령은 단어 뒤에 이어붙는 반점, 온점, 그밖의 문장부호 뒤에 위치해야 한다.

숫자 대신 기호 문자나 임의의 텍스트를 각주표지로 쓰고 싶다면 `\footnote` 명령에 관련된 카운터를 재정의한다.

```
\renewcommand{\thefootnote}{read me!}
This footnote\footnote
{I mean this one.}
says it all.
```

This footnote[read me!] says it all.

[read me!] I mean this one.

이런 방식으로 각주 번호를 로마 숫자나 멋진 기호문자로 바꿀 수 있다.

```
\renewcommand{\thefootnote}
{\Roman{footnote}}
This\footnote{The first.}
is the first footnote,
and this\footnote{The second.}
is the second.
\renewcommand{\thefootnote}
{\fnsymbol{footnote}}
The end.\footnote[8]{At last!}
```

This[II] is the first footnote, and this[II] is the second. The end.[†]

[II] The first.

[II] The second.

[†] At last!

`\fnsymbol{footnote}`라는 것을 잘 살펴보자. 이것은 각주 번호를 나타내는 카운터 `footnote`의 값 1···9에 각각 할당된 아홉 개의 기호표지이다. * † ‡ § ¶ ‖ ** †† ‡‡. 동일한 각주에 대해서 여러 번 참조하려면 각주 숫자를 하나하나 써넣지 말고 다음과 같이 하라.

```
This\footnote{the first.}
\newcounter{myfootnote}
\setcounter{myfootnote}
  {\value{footnote}}
and that\footnote{the second.}
are footnotes: please read note
\footnotemark
[\value{myfootnote}] again.
```

This[1] and that[2] are footnotes: please read note [2] again.

[1] the first.

[2] the second.

주의: minipage 안에서는 `mpfootnote`와 `thempfootnote`라는 별도의 각주 카운터로 번호가 매겨진다.

미주

각주에 관한 문제는 제법 복잡해서, 복잡한 각주, 둘 이상의 번호 체계를 갖는 복합각주, 각주 안의 verbatim, 각주 안의 수식, parbox안의 각주, 페이지별 각주번호 등의 문제를 해결하는 패키지들이 많이 있다. 이럴 경우 예를 들면 footnote나, footmisc 등의 사용설명서를 참고해야 한다. 그러나 LaTeX에서 각주 문제는 거의 대부분이 해결되어 있으므로 문제는 어떤 패키지를 참고해야 하느냐일 뿐이다. 가장 좋은 방법은 질문하는 것이다.

endnotes 패키지는 모든 각주를 문서 끝으로 몰아준다. preamble에 다음 한 줄을 추가해야 한다.

```
\let\footnote=\endnote
```

그리고 다음 몇 줄을 문서의 마지막에 둔다.

```
\newpage
\begingroup
\parindent 0pt
\parskip 2ex
\def\enotesize{\normalsize}
\theendnotes
\endgroup
```

enotez와 mbenotes가 endnotes와 비슷한 기능을 제공한다.

이 이외의 다른 명령도 있다. `endnotes.sty` 소스 파일을 읽어보라.

4.6 Insert/Indices 차례

차례, 표 차례, 그림 차례를 생성하고 넣는 것은 LaTeX에서 아주 간단한 일이다. 다음 몇 줄을 첫번째 `\section`이나 `\chapter`보다 이전에 써주기만 하면 된다.

차례의 점선이나 페이지 표시 방법 등을 사용자가 수정하려면 tocloft 패키지를 이용할 수 있다. 그리고 float 패키지를 이용하여 새로운 플로트를 정의할 때 그 목록을 만들 수도 있다. 이 두 가지 패키지는 모두 memoir에 이미 들어 있다.

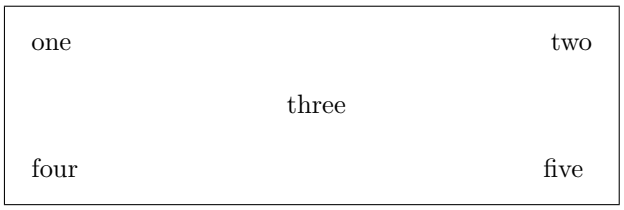
```
\tableofcontents
\listoffigures
\listoftables
```

4.7 Insert/Vertical and Horizontal Space 간격과 공백

이 항목에 해당하는 기능이 존재하는 워드 프로세서는 내가 아는 한 없다. LaTeX은 이 일을 아주 우아한 방식으로 처리한다.

공백 채우기는 텍스트를 수직으로 수평으로 가운데 두기 위해 사용한다. 워드 프로세서에서는 이것이 매우 어려운 일이다. 여러 번 시행착오를 거쳐야만 겨우 비슷하게 된다. `\null`이나 `~`를 고정점으로 사용하고 그 뒤에 이어 `\vfill`이나 `\hfill`을 다음 보기와 같이 써보라.

```
1  one \hfill two\\
2  \vfill
3  ~ \hfill three \hfill ~\\
4  \vfill
5  four \hfill five
6  \null
```



이런 식으로 하지 않고 텍스트를 페이지의 절대 위치에 고정시키고 싶은 경우에는 textpos나 pgf를 이용할 수 있다.

보통 LaTeX은 사용자가 마음대로 빈 공간을 넣는 것을 허락하지 않는다. 소스에서 스페이스를 두 번 친다고 해서 출력물에서 두 칸의 스페이스가 나타나는 것은 아니다. 그러나 문서가 엉망이 되어도 상관없다면 잘라지지 않는 공백 기호 ~(tilde)를 두 번 써보라. 실제로 출력에도 두 개의 공백이 찍힐 것이다.

또, `\hspace`를 다음과 같이 사용할 수 있다.

```
This is a \hspace{2cm}
2-cm-wide hole.
```

This is a 2-cm-wide hole.

\hspace는 '앞 글자'가 있어야 동작한다. 왼쪽 끝에서 \hspace는 아무 의미가 없을 것이다. 이럴 때도 강제로 간격을 주려 한다면 \null\hspace와 같이 하거나 또는 '별표 붙은 명령' \hspace*를 쓸 수 있다.

4.8 Insert/Tabs 탭

tabbing 환경은 TAB 키의 동작과 거의 비슷한 기능을 제공하여 종으로 가지런한 텍스트 정렬을 하게 해준다. 이 환경에서 사용되는 명령을 요약하면 다음과 같다.

Command	Action
\=	Sets a tab stop
\>	Advances to the next tab stop
\+	Sets the left margin one tab stop to the right
\-	Sets the left margin one tab stop to the left
\\	Ends a line
\pushtabs	Saves all tab stop positions
\poptabs	Restores previously saved tab stop positions

실제로 사용되는 예를 보인다.

```
\begin{tabbing}
% let's set the tab positions
~ \hspace{1cm} \= ~ \hspace{2cm} \=
~ \hspace{3cm} \= \kill % discard text
Zero \> One \> Two \> Three \\
Zero \> One \> \> Three \+ \\ % go right
Zero \> Two \> Three \- \\ % go left
Zero \> One \> Two \\
\pushtabs % save tab positions
new tab 1{\ldots} \= new tab 2 \\
new \> tab \\
\poptabs  % restore tab positions
Zero \> One \> Two \> Three
\end{tabbing}
```

Zero One Two Three
Zero One Three
Zero Two Three
Zero One Two
new tab 1⋯ new tab 2
new tab
Zero One Two Three

tabbing 환경이 일부 불편한 점이 있는 데 비해 tabto 패키지는 더 편리하게 tab을 이용한 정렬을 할 수 있게 해준다. 특히 특별히 특정 환경으로 둘러싸지 않아도 tab을 쓸 수 있다는 것이 장점이다. 해당 패키지의 문서를 참고하라.

또 tabular와 table 환경을 참고하라.

4.9 Insert/Cross Reference 교차참조

\label, \ref, \pageref 명령만 있으면 텍스트에 레이블을 달고 그것에 대해 교차참조할 수 있다. 레이블의 표준 포맷은 prefix:suffix 꼴인데, prefix는 cha(장), eq(수식), fig(그림), sec(절), tab(표)와 같은 것이 될 수 있다. 이와 같이 prefix를 붙이는 것은 소스를 좀더 알아보기 쉽게 만들려는 것으로 꼭 이런 형식이어야만 하는 것은 아니다.

$\mathcal{AMS}$-math는 수식 번호를 괄호로 감싸주는 \eqref라는 명령을 별도로 제공한다.

절 번호, 표 번호, 그림 번호, 페이지 번호 등을 참조하려면 \label과 \ref 명령을 다음 보기와 같이 사용한다.

```
\paragraph{Example.}\label{par:example}
This paragraph appears
in Section~\ref{par:example}
on page~\pageref{par:example}.
```

Example. This paragraph appears in Section 4.9 on page 19.

[역자 보충] \label을 붙여서 나중에 \ref할 수 있는 '번호'에는, 장절번호, 표와 그림의 번호, 나열 환경의 아이템 번호 등이 있다. 각각 \section류 명령, \caption 명령, \item 명령 직후에 \label을 붙여야 한다. 특히 표와 그림은 반드시 \caption 이후라야 번호가 제대로 동작한다는 사실을 알아두자.

물론 사용자가 prefix를 마음대로 정해도 된다. 다음 보기는 enumerate에 적용한 예이다.

```
\begin{enumerate}
  \item{first step: skip to
  \ref{item:end} \label{item:start}}
  \item{another step (unreferenced)}
  \item{end: go back to
  \ref{item:start} \label{item:end}}
\end{enumerate}
```

1. first step: skip to 3
2. another step (unreferenced)
3. end: go back to 1

4.10 Insert/Margin Notes 방주

정말 쉽다. \marginpar{text}.

마진노트를 이용하여 이른바 '변2단' 조판을 하려고 할 때, 실제로 마진 문단의 처리가 아주 쉽지만은 않다. 방주 관련하여 sidenotes, snotez, marginnote 패키지들을 참고하라. 이 가운데 많은 기능이 memoir에 이미 구현되어 있다.

4.11 Insert/Text Frame 프레임친 문단

포스터나 광고판을 조판한다고 생각해보자. 특정 텍스트나 그림을 페이지상의 고정된 위치에 갖다두어야 할 때가 있다. 이럴 경우 textpos패키지를 이용한다. 샘플이 그림 A.5에 있다. (부록 A를 보라.)

유명한 레이아웃 프로세서에서 프레임이란 페이지 상에 놓이는 특정의 구역을 말한다. 이것은 워드 프로세서에는 없는 개념이다. LaTeX으로 이와 같은 레이아웃 프로세서의 프레임을 구현한 예로는 flowfram 패키지가 있다. 이것은 이미 텍스트의 흐름을 조판하는 것이 아니므로 대부분의 문서에서는 볼 수 없는 것이다.

한편, 프레임을 단순히 박스쳐진 텍스트로 부르는 경우도 있는데 이것은 framed나 boxedminipage, 또는 boites 패키지에 의하여 구현이 가능하다. 특히 framed, boites는 페이지 사이가 나누어지는 프레임쳐진 문단을 식자할 수 있게 해주기도 한다. mdframed나 tcolorbox는 좀더 화려한 효과를 가지는 프레임친 문단의 구현을 가능하게 한다.

꼭 특정한 위치에 가져다두어야 하는 것이 아니라면 minipage (miniature page) 환경을 사용하라. 이 텍스트는 minipage 안에 식자한 것이다…

…그리고 boxedminipage 환경으로 둘러싸면 minipage에 프레임을 쳐서 보여준다. 같은 이름의 패키지를 로드하여야 한다.

minipage 환경은 다음과 같은 방식으로 사용한다.

```
\begin{minipage}[position]{width}
...
\end{minipage}
```

boxedminipage에서 텍스트와 프레임 사이의 간격 설정은 다음과 같이 한다.

```
\setlength{\fboxsep}{5mm}
```

4.12 Insert/Figure 그림넣기

(LaTeX에서 그림을 포함하는 문제에 대한 안내서로 'Using Imported Graphics in LaTeX 2ε', a.k.a. `epslatex.ps`가 있다.)

좀 오래된 안내서이고 요즘 사정과 달리 eps 그림에 대해 집중적으로 설명하고 있지만 훌륭한 문서.

'figure'라 함은 비단 그림 파일만을 의미하는 것이 아니라 텍스트, 표 등 `figure` 환경 안에 놓을 수 있는 것은 뭐든지 상관없다. 다음 보기를 보자.

```
\begin{figure}[htbp]
% [htbp] specifies the
% preferred placement: here, top,
% bottom, or separate page.
  \begin{center}
  \texttt{=8-)}
  \end{center}
  \caption{A smiley representing
  the author of this guide.}
  \label{fig:mysmiley}
\end{figure}
```

=8-)

그림 1: 저자를 표현한 스마일리

그림들이 figure 관련 코드를 작성한 바로 그 위치에 정확하게 나타난다는 보장이 없음에 주의하자. 사실 워드 프로세서와 LaTeX의 가장 중요한 차이 중 하나가 그림들이 고정된 위치를 갖지 않는다는 점이다. 그림은 LaTeX이 스스로 결정하는 최적의 위치로 '떠다닌다'. 그러므로 문장을 쓸 때는 '아래 그림'이나 '위의 그림'과 같이 써서는 안 되고 '그림~`\ref{fig:label}`'과 같이 작성해야 한다. 그림이 어느 위치에 올지 모르기 때문이다.

사실 LaTeX 처음사용자가 가장 짜증스러워하는 것이 표나 그림이 "제 위치"에 오지 않는다는 것이다. 그러나 다시 생각해보면, 과연 사용자가 원하는 그 위치가 반드시 최적의 위치일까? 그림이 다음 페이지로 넘어가고 앞 페이지의 아래쪽이 휑하게 비는 것이 꼭 좋은 위치인 것일까? 이것은 짜증을 부릴 일이 아니라 왜 LaTeX이 표나 그림을 그런 식으로 처리하는지 이해하는 것이 더 낫지 않을까 싶다. 만약 그래도 표/그림의 위치를 LaTeX에게 맡기기 싫다면, 그냥 float 환경 안에 넣지 않으면 된다.

이런 속성 때문에 그림이나 표를 **떠다니는 개체**라고 부른다. 특정의 표나 그림이 정확하게 어떤 위치에 있어야 할 이유가 꼭 있다면, here 패키지를 사용하라. 이 패키지는 위지 지정 인자로 `H`('바로 여기(HERE)!'라는 뜻이다)를 제공한다.

이 문서에서는 here 패키지를 언급하고 있다. 일반적으로 float 패키지의 `[H]` 위치지정자를 쓰는 것이 좋다.

Encapsulated PostScript (`.eps`) 포맷의 그림이 하나 있다고 하자. 이 그림을 LaTeX 소스 파일에 삽입하려면 graphicx 패키지와 그림 2에 보인 것과 같은 명령을 이용한다.

```
\begin{figure}
\begin{center}
\fbox{\includegraphics
[width=0.5\textwidth, angle=-90]
{gnuplot.ps}}
\caption{A Gnuplot graph.}
\label{fig:gnuplot}
\end{center}
\end{figure}
```

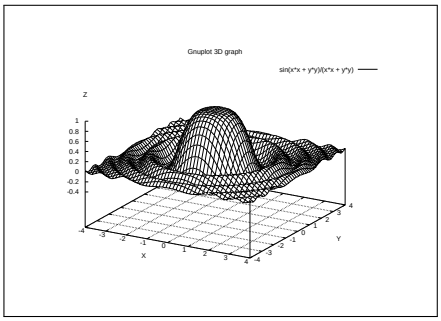


그림 2: Gnuplot으로 그린 그래프

`latex`과 `dvips`로 문서를 조판할 때는 EPS 파일만이 동작한다. 반면 `pdflatex`은 JPG, PNG, (당연히) PDF 파일을 받아들인다. `pdflatex`에서 PDF, PNG, JPG 그림을 쓰는 방법이 일반 사용자에게 더 편리하다.

`pdflatex`은 EPS를 적절하게 자동으로 변환하여 처리한다. XeLaTeX, LuaLaTeX도 마찬가지이다.

일반 그래픽 포맷(`.jpg`, `.gif`, `.png` 등)을 `.eps`로 변환하는 패키지들이 몇 가지 있다. 예를 들면 ImageMagick (http://www.imagemagick.org), GIMP (http://www.gimp.org) 등. 그러나 이런 응용 프로그램들은 엄청난 크기의 PostScript 파일을 만들어낸다.

제일 좋은 것은 비트맵을 압축하여 컴팩트한 PostScript 파일을 만들어내는 응용 프로그램을 이용하는 것이다. jpeg2ps (http://www.pdflib.com/jpeg2ps/index.html), bmeps (CTAN://support/bmeps)와 같은 유틸리티가 좋다. 앞의 것은 .jpg 파일을 내장하는 데 제일 낫다고 하고, 뒤의 것은 다양한 그래픽 포맷을 지원한다.

같은 소스에서 .pdf와 .ps를 모두 만들려 한다면 다음과 같은 코드를 포함하는 방법이 있다.

현재, 그림 포맷과 관련해서는 다음과 같이 알아두면 된다. latex-dvips를 쓰는 경우에만 오직 .eps 그림이 필요하다. 그 외의 경우—PDFLaTeX, XeLaTeX, latex-dvipdfmx 등—.png, .jpg, .pdf, .eps 어떤 포맷이라도 상관없다.

```
\usepackage{ifpdf}
...
% include the right options
\ifpdf
  \usepackage[pdftex]{graphicx}
  \pdfcompresslevel=9
\else
  \usepackage{graphicx}
\fi
...
% include the right graphic file
\ifpdf
  \includegraphics{file.png}
\else
  \includegraphics{file.eps}
\fi
```

\newpage와 \clearpage의 차이를 알 수 있다. 후자는 현재까지 처리되지 않은 모든 플로트를 모두 식자한 다음 새로운 페이지를 생성한다.

☞ 18개 이상의 플로트가 처리되지 않은 상태로 대기중이면 'Too many unprocessed floats'라는 에러를 만나게 된다. 이 문제를 해결하는 가장 빠른 방법은 \clearpage를 서너 개의 그림 사이에 넣어주는 것이다. 또는 morefloats 패키지를 이용할 수도 있다.

문단을 파고드는 그림

잡지 등의 레이아웃에서 볼 수 있는 것으로 그림이 텍스트 문단을 파고 들어가도록 배치하는 것이 있다. wrapfig 패키지를 이용하면 다음과 같은 모양을 만들 수 있다. 다음 보기에서 인자를 준 방식을 유심히 보도록 하자. 각각 문단에서 좁아지는 행의 수, 그림의 위치(r 또는 l), 그림 걸이 길이(overhang), 그림의 폭(width)이다. 이 가운데 첫째, 셋째 인자는 옵션 인자이므로 생략할 수 있다.

```
1  If you meet this guy, give him some money.
2  \begin{wrapfigure}[4]{l}[5pt]{2cm}
3  {\Huge
4   \texttt{=8-)}
5  }
6  \end{wrapfigure}
7  The reason may not be apparent to you,
8  but I can assure that your money
9  will end up in good hands.
10 I say again, if you meet this guy,
11 give him some money: he knows how to
12 use it properly. OK?
```

If you meet this guy, give him some money.

=8-)

The reason may not be apparent to you, but I can assure that your money will end up in good hands. I say again, if you meet this guy, give him some money: he knows how to use it properly. OK?

4.13 Insert/Shapes 도형 그리기

LaTeX 자체에 `picture` 환경이 있어서 이를 이용하여 `\circle`, `\oval` 등의 명령으로 도형을 그릴 수 있다. 내 생각에, GUI 없이 텍스트로 그림을 그리는 것은 너무 어렵고 `picture` 환경 자체가 한계가 많다. 따라서 훌륭한 그림그리기 프로그램을 이용하는 것이 훨씬 낫다. 그 중에 무료이고 오픈소스인 벡터 그리기 프로그램 Inkscape (https://inkscape.org/)와 Pstoedit (http://www.pstoedit.net/)이 있다.

Inkscape를 실행하고 도구를 이용하여 원하는 도형을 그린다. LaTeX 텍스트를 삽입하려면 Extensions/Render/LaTeX formular... 메뉴를 선택하여 텍스트를 적어넣고 Apply를 클릭하면 된다. 그림 3을 보라.

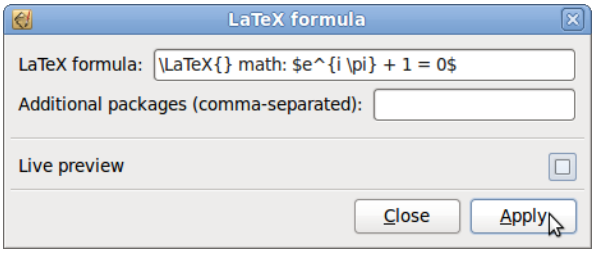


그림 3: LaTeX 수식 삽입

LaTeX 생성된 텍스트는 그래픽 개체로 포함되므로 원하는 대로 편집할 수 있다. 결과 그림은 LaTeX에 친숙한 그림 포맷, PDF, PNG 등으로 내보내기할 수 있다. 더 자세한 것은 다음을 참고하라. http://www.ctan.org/tex-archive/info/svg-inkscape.

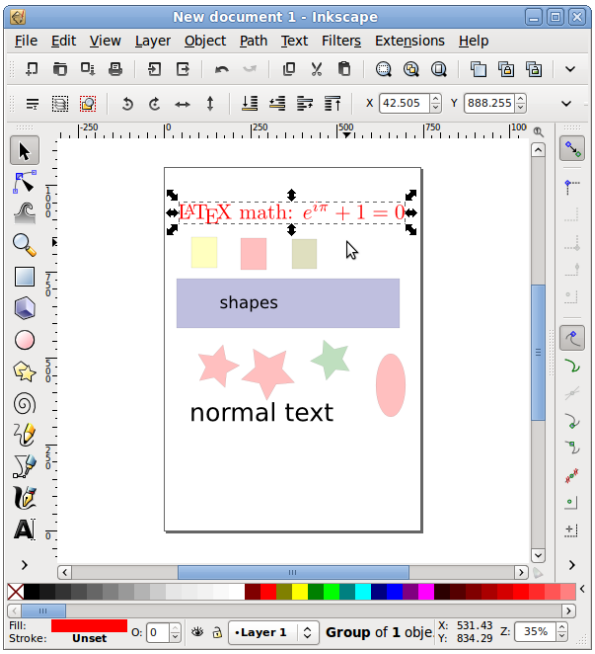


그림 4: LaTeX 개체도 편집 가능

뭔가 더 대단한 것을 해보고 싶다면 다음 패키지나 프로그램에 대해 알아보기 바란다.

- pgf 는 그림 그리기 TeX 매크로 패키지이다.
  http://sourceforge.net/projects/pgf/

- GLE (Graphics Layout Engine) 는 출판물 품위의 그래프, 플로트, 다이어그램, 그림과 슬라이드를 제작하기 위한 그래픽 스크립팅 언어이다.
  http://www.gle-graphics.org
- asymptote는 강력한 벡터 그래픽 기술 언어이다. 이공학 분야 드로잉을 그릴 수 있는 자연스러운 좌표기반 프레임웍을 제공한다.
  http://asymptote.sourceforge.net/
- ePiX는 GNU/Linux와 그 친척 플랫폼에서 수학적으로 정확한 그림, 플로트, 동영상을 제작하기 위한 쉬운 문법으로 된 배치 유틸리티 모음이다.
  http://mathcs.holycross.edu/~ahwang/current/ePiX.html;
- pstricks는 LaTeX 문서에서 Postscript 그림을 포함하는 매크로 모음이다.
  http://tug.org/PSTricks/main.cgi/

이 패키지들은 출판물-품위의 PostScript 그림을 LaTeX에서 활용할 수 있게 해준다. 이밖에도 많으므로, "LaTeX vecter graphics"로 웹 검색을 해보기 바란다.

삽입할 수 있는 '도형'의 모양도 여러 가지 있다. 입맛에 맞는 것을 찾아보려면 TeX Showcase 페이지로 가 보라. http://www.tug.org/texshowcase/.

4.14 Insert/Line 괘선

임의의 길이와 굵기로 된 선분을 그리는 명령은 `\rule`이다. 이 명령은 두 개의 인자를 취하는데 각각 선의 길이와 높이를 나타낸다.

```
1  This is a page-wide
2  rule:\\
3  \rule{\linewidth}{1pt}
4  but this one is shorter
5  and thicker:\\
6  \rule{2cm}{2mm}
```

This is a page-wide rule:

but this one is shorter and thicker:

`\dotfill`은 점으로 만들어진 재미있는 '선'을 그어준다. 이것은 서로 연관된 것들을 이어주는 데 가끔 쓰인다.

```
1  Total price \dotfill \euro~10
```

Total price ..... € 10

4.15 Insert/Hyperlink 하이퍼링크

hyperref 패키지는 URL 및 기타 외부 참조를 가능하게 한다. `dvipdf`나 `pdflatex`과 함께 쓰면 브라우징 가능한 `.pdf` 문서를 만들어낸다. 예를 들면, 이 문서는 다음과 같이 선언하여 제작하였다.

```
\usepackage[colorlinks,
        urlcolor=blue,
```

```
filecolor=magenta,
linkcolor=darkred,
hyperfootnotes=false]{hyperref}
```

예를 들어보자.

```
1  The \hypertarget{ctan}{CTAN} main site
2  is \url{http://www.ctan.org}, a.k.a
3  \href{http://www.ctan.org}{CTAN://}.
4
5  Listen to \href{run:midifile.mid}
6  {this MIDI file}.
7
8  Click \hyperlink{ctan}{here} to go
9  back to the top.
```

The CTAN main site is http://www.ctan.org, a.k.a CTAN://.
Listen to this MIDI file.
Click here to go back to the top.

보는 바와 같이, `\url` 명령은 monospace 폰트로 식자한다. 본문 텍스트와 같은 폰트를 쓰려면 다음 영령을 `\hyperref` 선언 이후에 둔다.

```
\urlstyle{same}
```

`\hypertarget`과 `\hyperlink` 명령은 HTML에서와 같은 내부 링크를 만들어준다. `\href`는 URL이나 외부 파일에 대한 링크를 만든다. `run:` 파라미터에 주의하라. 이것은 멀티미디어 플레이어, 오피스 프로그램 등과 같은 외부 프로그램을 실행시켜준다. 내가 아는 한 이 기능은 오직 Adobe Reader, Okular, Evince에서만 동작한다.

Linux나 다른 UNIX 계열 운영체제에서는 자신의 PDF 리더에서 외부 파일이 참조되었을 때 어떤 프로그램을 실행해야 할지를 알려주어야 한다. 자신의 `.mailcap`이나 `/etc/mailcap`에 다음 내용을 써넣으면 된다.

```
audio/midi;/usr/bin/timidity %s
audio/*; xmms %s
video/*; xine -pfhq %s
```

예를 들면 외부 파일의 타겟을 참조하거나, pdf bookmark를 만들거나 Adobe Acrobat의 메뉴를 조작하는 등의 작업도 가능하다.

hyperref의 패키지 문서를 읽어보면 더 많은 기능에 대해 알 수 있고 예제를 볼 수 있다.

4.16 Insert/Comment 주석문

각 행 앞에 % 기호를 붙이면 그 행은 주석문이 되어 문서의 출력에 반영되지 않는다. 여기에 출력물에는 보이지 않는 저자 자신의 메모나 노트를 기록할 수 있다. 아니면, comment 패키지를 통하여 `comment` 환경을 써서 문서의 일부를 무시하게 만들 수 있다.

5 Format 포맷 메뉴

일반적으로 문서의 주요 포맷 설정은 `\documentclass`의 파라미터로 지정한다. 기본 글꼴 크기(10, 11, 12pt), 용지(`a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper`, `executivepaper`), 방향(`portrait`, `landscape`) 등.

```
\documentclass[a5paper,landscape,12pt]{article}
```

위의 세 가지 이외의 폰트 크기를 지정하는 것도 가능한데 이에 대해서는 5.2절에서 설명한다.

5.1 Format/Line Spacing 행간격

setspace 패키지는 `singlespace`, `onehalfspace`, `doublespace` 환경을 제공한다. 그리고 `\spacing{amount}` 명령(환경)은 주어진 크기만큼 행간격을 설정해준다.

memoir에는 setspace의 기능이 그대로 들어 있으나 명령과 환경의 첫 글자가 대문자로 시작하는 것이 다르다. 예를 들면 `\Spacing`.

```
\begin{spacing}{2.5}
These two lines \\
are crazily spaced!
\end{spacing}
\begin{spacing}{1}
Much better, these lines\\
have a pretty space.
\end{spacing}
```

These two lines

are crazily spaced!

Much better, these lines
have a pretty space.

전체 문서의 행간격을 적용하려면 `\linespread{factor}` 명령을 preamble에서 사용한다. 기본값은 `factor` = 1로 되어 있다. 이 값이 커지면 행간격도 커진다. 1.6이면 대략 double line spacing에 해당한다.

한글 문서는 기본값이 1.333

5.2 Format/Character 글자 모양

표준적인 글자 속성은 표 3에 열거하였다. 글자 크기는 표 4를 보라. 실제 폰트 크기는 `documentclass` (10, 11, 12pt)의 기본 폰트 사이즈에 따라 달라진다는 점에 주의하라. 표 5를 볼 것.

이탤릭체와 강조체 사이의 차이에 주의하라. `\emph` 명령은 이탤릭체 안에서는 upright 체로 변한다. *For example, this portion of text is typeset in italics, and* these words *are emphasised in upright.* `\emph` 명령은 타이포그래피 지정 명령이 아니라 '강조'를 의미하는 논리적인 명령임을 알 수 있다.

[역자 보충] 한글 문서에서 `\emph`를 주었을 때, 일반적으로 *기울어진 글자*로 나타날 수 있다. 또는 **폰트 대치** 방식으로도 구현할 수 있다. 그러나 상황에 따라서 이것이 적절하지 않을 수도 있고 (XeLaTeX문서에서는) 폰트 설정 명령에 따라 아무런 변화가 없을 수도 있다. 강조를 처리하는 방법에 대해서는 `kotexdoc` 및 `xetexko-doc`을 참고하라. 한편, `\dotemph`로 글자 위에 점을 찍어서 강조를 표현하는 방법도 제공한다. 한글 문서에서 강조의 구현은 사용자가 세심하게 선택해야 하는 문제이다.

또한 아래첨자는 수학 모드 안에서만 동작한다는 것도 알아두자. 일반 텍스트에서 첨자를 사용하는 트릭은 다음과 같다.

`\textsuperscript` 명령을 통해서 위첨자는 텍스트 모드에서도 쉽게 식자할 수 있다. 텍스트의 아래첨자도 `\textsubscript` 명령을 쓸 수 있는데 이것은 별도로 정의된 패키지가 사용되어야 한다. XeLaTeX에서는 xltxtra 패키지가 이 명령을 정의한다.

```
this is
$_{\mbox{\footnotesize{subscript}}}$
```

this is $_{\mbox{\footnotesize{subscript}}}$

Text attribute	Environment form	Example
\textnormal	textnormal	main document font
\textrm	rmfamily	roman
\textit	itshape	*italics*
\emph	n/a	*emphasis*
\textmd	mdseries	medium weight (default)
\textbf	bfseries	**boldface**
\textup	upshape	upright (default)
\textsl	slshape	*slanted*
\textsf	sffamily	sans serif
\textsc	scshape	SMALL CAPS
\texttt	ttfamily	typewriter
\underline	underline	underline
\textsuperscript	n/a	this is superscript
\mathrm	n/a	$x^n + y^n \neq z^n \forall n \neq 2$
\mathbf	n/a	$\mathbf{x^n + y^n \neq z^n \forall n \neq 2}$
\mathsf	n/a	$\mathsf{x^n + y^n \neq z^n \forall n \neq 2}$
\mathtt	n/a	$\mathtt{x^n + y^n \neq z^n \forall n \neq 2}$
\mathit	n/a	$x^n + y^n \neq z^n \forall n \neq 2$
\mathnormal	n/a	$x^n + y^n \neq z^n \forall n \neq 2$
\mathcal	n/a	$\S\backslash + \dagger\backslash \neq \ddagger\backslash \forall \backslash \neq \in$

표 3: 글자 모양 선택 명령

Font size	Example
tiny	sample text
scriptsize	sample text
footnotesize	sample text
small	sample text
normalsize	sample text
large	sample text
Large	sample text
LARGE	sample text
huge	sample text
Huge	sample text

표 4: 폰트 크기

화학식의 첨자

대부분의 화학식은 수식 모드로 입력한다. 첨자는 ^와 _로 얻는다. mhchem 패키지가 더 간단한 방법을 제공하는데, 숫자는 무조건 하첨자로 식자하고 상첨자가 되어야 할 때 ^를 쓰는 것이다. 화학식은 \ce 명령으로 감싼다.

```
\ce{H2O + CO2 -> H2CO3}\\
\ce{CaCO3 -> Ca^2+ + CO3^2-}\\
\ce{CO3^2- + H2CO3 -> 2 HCO3^-}\\
\ce{CaCO3 + H2CO3 -> Ca^2+ + 2HCO3^-}
```

$$\begin{aligned}
&H_2O + CO_2 \longrightarrow H_2CO_3\\
&CaCO_3 \longrightarrow Ca^{2+} + {CO_3}^{2-}\\
&{CO_3}^{2-} + H_2CO_3 \longrightarrow 2{HCO_3}^-\\
&CaCO_3 + H_2CO_3 \longrightarrow Ca^{2+} + 2\,{HCO_3}^-
\end{aligned}$$

Default font size	10pt	11pt	12pt
tiny	5	6	6
scriptsize	7	8	8
footnotesize	8	9	10
small	9	10	10.95
normalsize	10	10.95	12
large	12	12	14.4
Large	14.4	14.4	17.28
LARGE	17.2	17.28	20.74
huge	20.7	20.74	24.88
Huge	24.8	24.88	24.88

표 5: 글자의 실체 크기

밑줄긋기

일반적으로 밑줄 긋기는 사용하지 않는다. 이것은 옛날 텔레타이프 시대의 유물로서 보기에 좋지도 않다. 정말로 밑줄을 긋지 않으면 안 되는 경우, 몇 가지 재미있는 밑줄 스타일도 제공하는 ulem 패키지를 사용할 수 있다. 다음 보기를 보자.

xetexko에서 ulem 스타일의 밑줄을 그으려면 패키지를 로드해야 한다. 그러나 luatexko는 이 기능을 이미 구현하고 있으므로 별도로 ulem을 선언할 필요 없다.

```
1  \uline{important}
2  \uuline{urgent}
3  \uwave{boat}
4  \sout{wrong}
5  \xout{removed}
```

important urgent boat ~~wrong~~ ~~removed~~

주의: ulem은 \emph 명령을 재정의하여 밑줄 긋기로 바꾼다. 이것을 피하려면 패키지를 다음과 같이 불러야 한다.

```
\usepackage[normalem]{ulem}
```

Format/Character Size 글자 크기

표준 폰트 사이즈로 충분치 않다면 extsizes가 도움이 된다. 표준 문서 클래스의 글자 크기 옵션을 '확장'하여 8–12, 14, 17, 20pt 옵션을 추가해준다.

예를 들어 어떤 글을 본문 17포인트로 조판하기를 원한다고 하자. preamble에 다음과 같이 쓰면 된다.

```
\documentclass[17pt]{extarticle}
```

큰 글자를 얻는 또다른 방법은 type1cm 패키지를 이용하는 것이다. 다음과 같은 명령이 가능하다.

type1cm을 선언해야 하는 것은 예전 pk 폰트를 활용하던 때 필요한 것이었다. 현재의 현대적 TEX 시스템에서는 대부분 불필요하다.

```
\fontsize{72pt}{72pt}\selectfont
No Smoking
```

(이 샘플은 너무 커서 이 페이지에 맞추기가 어렵기 때문에 결과를 보이지 못한다.)

인자 두 개는 각각 폰트의 사이즈와 베이스라인스킵(\baselineskip)의 크기이다.

또다른 방법으로 \resizebox를 이용하는 것이 있다. 다음과 같다.

```
\resizebox{!}{1cm}{1-cm tall}
```

1-cm tall

문단 첫 글자를 크게 만들어서 몇 줄에 걸쳐 식자하는 것은 lettrine 패키지를 이용한다. 이 패키지가 제공하는 \lettrine 명령은 사용자 설정이 가능한데, 다음 보기는 기본 동작이다.

```
\lettrine{D}{ropped} capitals
at the start of a paragraph
can be obtained using the
lettrine package.
```

DROPPED capitals at the start of a paragraph can be obtained using the lettrine package.

이 부분은 XeTeX이 등장하기 전 legacy TeX의 폰트와 LaTeX 2ε의 NFSS에 대해서 기술하고 있다. XeTeX의 등장으로 사정이 무척 많이 달라졌다. 이에 대해서 역자가 말미에 짧은 한 문단을 추가한다.

Format/Character Font 폰트

LaTeX은 자체 폰트를 사용하는데 필요하면 자동으로 METAFONT 시스템에 의하여 생성한다. 이렇게 하면 이식성이 보장되며 매우 좋은 출력 품질을 얻을 수 있다. 그러나 우리는 Times, Helvetica, Sans Serif…와 같은 다른 폰트에 익숙하다.

다행히 LaTeX은 PostScript 폰트를 사용할 수 있다. 다음 패키지들은 익숙한 PostScript 폰트를 사용할 수 있게 하는 것들이다. avant, avangar, bookman, chancery, charter, courier, helvet, helvetic, ncntrsbk, newcent, palatcm, palatino, pifont, times, utopia, zapfchan. 문서에 \usepackage{times}라고 써넣고 결과를 살펴보라. 유일한 주의사항은 LaTeX이 수식을 다룰 때 Computer Modern보다 나은 폰트가 없다는 점이다. PostScript 폰트를 수식에 썼을 때는 품질이 좀 떨어지는 것으로 보일 수도 있다.

위에 열거한 패키지들은 문서 전체의 폰트를 설정한다. PostScript 폰트를 텍스트 영역에만 적용하고자 한다면 폰트 패밀리를 아래 예와 같이 지정해주도록 하라. 일반적인 폰트 패밀리를 표 6에 열거하였다.

☞ 주의. 몇몇 폰트는 시스템에 따라 이용불가능할 수 있다.

```
This is Computer Modern Roman,
{\fontfamily{phv}\selectfont
this is Helvetica!}
```

다른 가능성은 표준 LaTeX 폰트를 PostScript 폰트로 교체하는 것이다. 예를 들면 Computer Modern Sans Serif가 나타날 위치의 폰트를 전부 Avantgarde로 바꾸는 것이다. 재정의 가능한 명령은 다음과 같고,

- \rmdefault (roman)
- \sfdefault (sans serif)
- \ttdefault (typewriter)

Family	Name
cmr	Computer Modern Roman
cmss	Computer Modern Sans Serif
cmtt	Computer Modern Typewriter
pag	Avantgarde
pbk	Bookman
phv	Helvetica
pnc	New Century Schoolbook
ppl	Palatino
ptm	Times
pcr	Courier

표 6: 일반적인 폰트 패밀리

- `\bfdefault` (boldface)
- `\mddefault` (medium)
- `\itdefault` (italics)
- `\sldefault` (slanted)
- `\scdefault` (small caps)
- `\updefault` (upright)

sffamily 기본 글꼴을 Computer Modern Sans Serif로부터 Avantgarde로 바꾸는 것은 다음과 같이 한다.

```
 % Avantgarde replaces sans serif
\renewcommand{\sfdefault}{pag}
```

[역자] XeTeX은 legacy TeX의 폰트 사용 방법에 일대 혁신을 가져왔다. Jonathan Kew가 제작한 XeTeX은 기존의 TeX 시스템에서 폰트를 처리하는 부분을 유니코드 폰트 사용 방식으로 교체한 것이다. 그 결과 운영체제에 설치된 오픈타입, 트루타입 폰트를 `.tfm`에 의존하지 않고 즉시 사용할 수 있게 되었다.

다음 보기는 본문에 나온 예를 살짝 수정한 것이다.

```
1  \fontspec{TeX Gyre Termes}%
2  This is TeX Gyre Termes,
3  \fontspec{TeX Gyre Heros}%
4  this is TeX Gyre Heros!
```

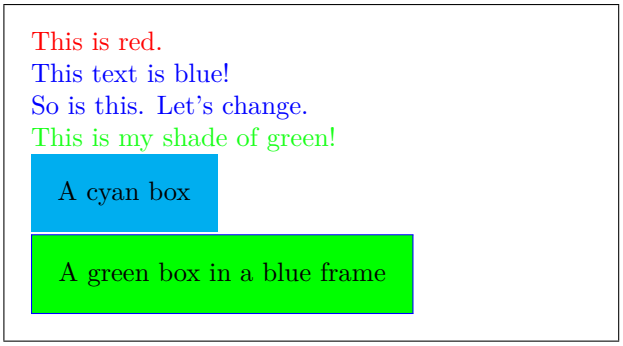
This is TeX Gyre Termes, this is TeX Gyre Heros!

물론 legacy TeX의 METAFONT와 LaTeX의 NFSS(우리에게 익숙한 폰트 선택 명령들)가 호환을 위해서 XeTeX에서도 동작하지만 오픈타입이나 트루타입을 사용하려면 fontspec 패키지의 `\fontspec` 명령을 익히는 것이 좋다. 역자는 본문의 '전통적인' 폰트 패밀리 선택 방식에 대한 설명을 수정하지 않고 그대로 보이면서 이 문단 하나를 추가한다. XeTeX의 폰트 선택 및 설정에 대한 간단한 예로 역자가 첨가한 부록 B절을 보라.

Format/Character Colour 색상

color 패키지를 이용하면 색상 이름과 적절한 명령을 사용할 수 있다. 미리 정의된 색상은 black, white, red, green, blue, cyan, magenta, yellow이다. 자신의 색상을 정의하는 것도 가능하다.

```
\textcolor{red}{This is red.}\\
\color{blue}
This text is blue!\\
So is this. Let's change.\\
\definecolor{mygreen}
{rgb}{0.1,1,0.1}
\color{mygreen}
This is my shade of green!\\
\color{black}
\colorbox{cyan}{A cyan box}\\
\fcolorbox{blue}{green}
{A green box in a blue frame}
```



이따금 페이지 전체에 배경을 깔고 싶을 때가 있다. 전에는 이 절차가 무척 복잡했지만 wallpaper 패키지로 매우 간단해졌다. 그림을 먼저 준비하고 이 패키지를 이용하면 손쉽게 배경에 그림을 놓을 수 있다.

앞서 지적한 바 있지만, \\는 문단 끝을 나타내는 기호가 아니다. 저자는 문단에 대해서 오해를 하고 있는 것 같은데, \\는 행의 줄바꿈을 강제하는 것일 뿐이고 문단을 끝내고 새 문단을 시작하는 것이 아닌 것이다. 문단 끝은 빈 줄 또는 \par가 오면 끝난다. 당연히 \\ 다음에 오는 행은 문단 첫 줄이 가지는 특성(들여쓰기 등)을 가지지 않는다. 새 문단이 아니기 때문이다.

\pagecolor라는 명령도 있다. 이 명령을 쓰면 어떤 일이 일어날까?

5.3 Format/Paragraph 문단 모양

LaTeX에 있어서 문단이란 무엇을 의미하는 것인지 상기하자. 문단이란 \\나 빈 줄로 끝나는 텍스트의 일부이다.

환경(*environments*)이라 하는 것은 텍스트의 일부분에 대해 정렬(alignment)이나 글꼴 선택과 같은 특정 속성을 부여하는 LaTeX의 방식이다. 이것은 마우스로 텍스트를 선택한 다음에 원하는 속성을 메뉴나 버튼 클릭으로 부여하는 것과 비슷하다. 중괄호로 텍스트 일부를 둘러싸는 것도 영역 선택과 비슷한 효과를 갖는다.

환경의 일반적 형태는 다음과 같다.

```
\begin{environment}
...text goes here...
\end{environment}
```

예를 들어 어떤 문단을 가운데 정렬하고 싶으면 center 환경을 사용한다.

```
\begin{center}
this text is centered
\end{center}
```

this text is centered

표준적인 환경들을 표 7에 열거하였다. 이어지는 절에서 언제 어떤 환경을 사용하는지 예를 들어보겠다.

Environment	Purpose
array	수식의 array
center	가운데 정렬
description	주제어 리스트 환경
enumerate	숫자 붙은 리스트 환경
eqnarray	여러 줄 수식
equation	수식 번호가 붙는 별행 수식
figure	그림 플로트 환경
flushleft	왼쪽 정렬
flushright	오른쪽 정렬
itemize	글머리 기호 붙은 리스트 환경
letter	Letters
list	리스트 환경 (다른 리스트 환경을 정의하는 데 사용)
minipage	미니페이지 (박스)
picture	선, 화살표, 원 등을 그릴 수 있는 그림 환경
quotation	긴 인용문. 둘 이상의 문단이 올 수 있음.
quote	짧은 인용문. 한 문단만 올 수 있음.
tabbing	tab 키를 사용하는 것과 같은 정렬 환경
table	표 플로트 환경
tabular	행과 열로 이루어진 표를 만드는 환경
thebibliography	참고문헌
theorem	Theorems, lemmas, etc
titlepage	수동 타이틀 페이지
verbatim	타자를 치는 것과 같은 입력한 대로 보이기 환경
verse	시나 가사 등 운문을 조판하는 환경

표 7: 표준 LaTeX 환경.

Paragraph/Horizontal Alignment 문단 정렬

정렬 방식의 더 정밀한 조절이 필요하거나 (이따금 그러하듯이) \justifying이 필요하다면 ragged2e 패키지를 사용한다.

텍스트는 좌우정렬되는 것이 기본이다. 왼쪽정렬, 오른쪽정렬, 가운데정렬 하기를 원할 때는 flushleft, flushright, center 환경을 쓴다. \raggedright, \raggedleft, \centering은 순서대로 각각의 환경에 대응하는 명령들이다. 그러나 이 명령은 새 문단을 시작하지 않는다.

한 가지 더 차이점을 언급하자면, 환경 (e.g., center) 은 명령 (e.g., \centering) 과 달리 새 문단을 시작하면서 환경의 앞뒤에 약간의 수직 간격을 준다. 이것은 매우 중요한 특성이다. 그리고 명령들은 일단 선언된 이후 계속해서 영향을 미치므로 명령이 유효한 범위 (scope) 를 {와 }로 잘 설정해주어야 한다.

Paragraph/Vertical Alignment 문단 사이 간격

문단 사이가 벌어지는 방식은 워드 프로세서 사용자를 종종 당황하게 만든다. **여러 개의 빈 줄과 여러 개의 스페이스는 한 개의 빈 줄이나 한 개의 공백과 똑같이 취급된다.** 따라서 소스에서 여러 개의 빈 줄을 넣는다고 해서 그만큼 문단 사이에 간격이 늘어나는 것이 아니다. 문단 사이 간격을 강제로 벌리려면 \smallskip, \medskip, \bigskip 명령을 사용해야 한다.

더 넓은 간격이 필요하다면 \vskip 명령을 다음 보기와 같이 사용한다.

\vskip 앞에 빈 줄 하나를 꼭 주어야 한다. 다른 수직 간격 명령들도 마찬가지.

```
1  These paragraphs will be
2  separated by 1.3 cm:\\
3  \vskip 1.3cm
4  there is a 1.3 cm gap above me.
```

These paragraphs will be separated by 1.3 cm:

there is a 1.3 cm gap above me.

\vskip 명령은 문단 사이에서만 동작한다는 것을 알아두자. 따라서 이전 문단이 없는 페이지의 제일 윗쪽에 이 명령이 오더라도 아무런 의미가 없다. 새 페이지를 시작하고 추가로, 예컨대 1.5cm를 남기고 싶을 때는 어떻게 할 것인가? 이럴 때는 \null 명령을 이용해서 텍스트에 '표지(mark)'를 설정한다.

```
\null
\vskip 1.3 cm
This text comes after 1.3 cm...
```

This text comes after 1.3 cm...

마지막으로 \vfill 명령은 두 문단 사이에 가변적인 빈 공간을 넣어서 두 번째 문단이 페이지의 하단에 맞추어지도록 한다. 예를 들면,

LaTeX에서 \vfil은 \vfill의 반(1/2)이다.

이 글에서는 \vspace 명령에 대해서 기술하고 있지 않다. \vspace는 간격을 중괄호로 둘러싸서 인자로 지시하지만 \vskip은 중괄호 없이 그냥 길이를 지정한다. 그리고 \vspace에는 별표붙인 명령 \vspace*가 있어서 페이지가 바뀌더라도 수직 간격 명령이 동작하게 할 수 있다.

```
This appears at the top of
the page{\ldots}
\vfill
{\ldots}and this at the bottom.
```

This appears at the top of the page…

…and this at the bottom.

Paragraph/Margins 여백

일반적으로 여백은 2.5절에서 본 바와 같이 문서 전체에 대해 설정되는 것이다. 일부 텍스트의 여백폭을 바꾸기 위해서 문서 중간에 이 파라미터 값을 바꾼다고 해도 동작하지 않는다. 한편 문단 폭이 달라지는 환경이 LaTeX에 이미 정의되어 있다. quote나 quotation 환경은 좌우 여백을 변경한다. 그런데 이런 환경을 쓰지 않고 임의로 문단 여백을 바꾸고 싶다면 다음 보기와 같이 새로운 환경을 만들어야 한다.

memoir와 changepage 패키지에는 문단 폭 변경을 위한 adjustwidth라는 환경이 있다. 흐르는 텍스트에는 대체로 잘 적용되지만 다른 리스트 환경과 함께 쓰면 원치 않는 동작을 하기도 한다.

```
\newenvironment{margins}[2]
{
  \begin{list}{} {
    \setlength{\leftmargin}{#1}
    \setlength{\rightmargin}{#2}
  } \item }
{\end{list}}
```

그런 다음에 이 새로운 환경을 사용한다.

```
As you can see, this paragraph
has normal margins.
\begin{margins}{0.5cm}{1cm}
 But please note that this
 paragraph has custom margins.
\end{margins}
```

As you can see, this paragraph has normal margins.

But please note that this paragraph has custom margins.

Paragraph/Indentation 들여쓰기

문단 첫 줄의 들여쓰기 크기를 설정하려면 \parindent 명령에 적당한 값을 준다. 다음 예에서 첫 줄 들여쓰기 값을 1cm로 설정하였다.

LaTeX의 기본 값은 1.5em 정도를 들여쓰는 것이다. 우리 글 문서에서는 그냥 1em(한 글자 폭)을 선호하는 경우도 많다.

```
\setlength{\parindent}{1cm}
```

\indent 명령과 \noindent 명령은 해당 문단의 첫 줄을 들여쓰기 하게/못하게 강제한다.

문단 첫 줄을 들여쓰기 하지 않고 문단 사이에 추가적인 간격을 주어서 구분하는 경우도 있다. 이런 식으로 하려면 문단 사이의 추가 간격 길이인 \parskip 값을 정해주어야 한다.

LaTeX은 장절 표제 다음의 첫 문단을 들여쓰기하지 않는 서양의 전통적 조판관행에 충실하다. 만약 첫 문단을 꼭 들여쓰기 하여야 한다면 indentfirst 패키지를 이용한다.

```
\setlength{\parskip}{3pt}
```

Paragraph/Border and Shade 문단 테두리와 음영

테두리쳐진 문단을 얻으려면 framed 패키지를 써서 할 수 있다.

\parbox를 \fbox로 감싸는 방법에 대한 언급이 있으나 번역본에서는 제외하였다.

```
1  \setlength{\FrameRule}{2pt}\setlength{\FrameSep}{5pt}
2  \begin{framed}
3      this is a framed paragraph!
4  \end{framed}
5  \definecolor{shadecolor}{rgb}{0.9,0.8,1}
6  \begin{shaded}
7      this is a shaded paragraph,
8      do you like it?
9  \end{shaded}
```

this is a framed paragraph!

this is a shaded paragraph, do you like it?

이 예에서 framed 환경은 문단을 테두리치는 것이고 shaded는 배경 음영을 넣는 것이다. shaded 환경은 shadecolor라는 색상을 사용하므로 미리 이 색상을 \definecolor해두어야 한다.

마찬가지로 boxedminipage 패키지의 같은 이름의 환경을 쓴다. 더 자세히 설명하자면 다음 명령

framed는 memoir에 이미 들어 있다. 별도로 \usepackage할 필요 없다. 이밖에 frame과 관련된 패키지로 efbox, mdframed, tcolorbox 등이 있다.

```
\framebox{
  \begin{minipage}[c]{\linewidth}
  text to be framed
  \end{minipage}
}
```

은 boxedminipage 환경과 동일하다.

\framebox 명령은 minipage의 폭을 옵션 인자로 부여할 수 있다.

\width는 minipage의 폭(width)을 텍스트가 차지하는 길이만큼으로 정한다. 당연히 원하는 길이를 직접 주어도 된다. \width를 이용하여 텍스트 폭에 맞추어서 프레임을 적용하는 방법을 보이겠다.

```
1  this is a
2  \framebox[\width]{framed}
3  word
```

this is a framed word

프레임의 폭을 조절하려면 파라미터를 수정한다. 두 번째 옵션 인자는 정렬 방식(여기서는 오른쪽)을 지정하는 것이다.

```
this is another
\framebox[2\width][r]{framed}
word
```

this is another framed word

Paragraph/Colour 문단 배경색

문단에 테두리를 치고 여기에 색을 입히려면 다음과 같이 한다.

```
\colorbox{yellow}{
  \begin{minipage}
  {0.8\linewidth}
  I am a minipage, my colour
  is yellow!
  \end{minipage}
}
```

I am a minipage, my colour is yellow!

예제로 보이기 위하여 행 길이의 80% 폭을 갖는 minipage에 색상을 칠하였다. 색상에 대해서는 5.2절을 볼 것.

소위 '형광펜 효과'로 soul 패키지의 \hl이 있다. 이것은 영문자의 경우 잘 동작한다. 한글에 대해서 이 비슷한 트릭을 만들어 본 예가 있다.

Format/Columns 다단

\twocolumn과 \onecolumn 명령은 새로운 페이지를 시작한 다음 그 이후 면에 대하여 2단 또는 1단으로 조판한다. 이 명령은 \documentclass의 옵션으로 사용할 수 있다. 이것으로 충분치 않다면 multicol 패키지가 제공하는 multicols 환경을 사용할 수 있다. 이 부분은 다음 명령으로 구현한 것이다.

multicols에 대해서는 저자의 착각인 듯. 패키지 이름은 multicol이고 환경의 이름이 multicols이다. 역자가 적절히 수정하였다.

```
\columnseprule=1pt
\begin{multicols}{2}[\subsection{\entry{Format}{Columns}}]
The commands \cmd{twocolumn} ...
\end{multicols}
```

단 사이의 간격은 \columnsep 파라미터로 제어가능하다. 단 사이에 긋는 구분선의 두께를 \columnseprule 길이값으로 설정할 수 있다. \twocolumn이나 \onecolumn 명령에 옵션 인자로 주어지는 텍스트는 2단 또는 1단이 시작하기 전에 그 다단 환경에 영향을 받지 않도록 식자된다.

5.4 Format/Styles 모양/스타일

이 소절은 역자가 추가한 것임

워드 프로세서 문서와는 달리 LaTeX은 처음부터 '구조화된' 문서를 만들게 되어 있다. 따라서 \chapter, \section, \subsection, \paragraph와 같은 장절 명령이 마련되어 있다.

보통 워드 프로세서에서 장절을 표현하기 위해서는 장절 타이틀에 해당하는 부분을 선택하여 글자 크기를 크게 하거나 두꺼운 글씨로 바꾸는 방법을 쓴다. 그리고 이렇게 만들어진 모양을 '스타일'로 등록하여 적용한다.

그렇다면 LaTeX에서 만약 `\section` 타이틀에 해당하는 모양을 바꾸고 싶다면 어떻게 하는가? 이 일을 해주는 패키지가 적어도 두 개 있다. 하나는 sectsty이라는 것이고 다른 하나는 titlesec이라는 것이다. 예를 들어 sectsty을 사용하는 경우, section의 글자를 sffamily(산세리프체)의 Large 크기로 하고 싶다고 하자. 그러면 다음과 같이 한다.

```
\sectionfont{\sffamily\Large}
```

또 한 가지 주제로 section에 붙는 카운터의 모양을 바꾸는 문제가 있다. 이 카운터는 LaTeX이 자동으로 붙여주기 때문에 편하지만 모양을 바꿀 필요가 있기도 하다. 이것은 별도의 패키지를 사용하지 않고도 예를 들어

```
\renewcommand\thesection{\arabic{section}}
```

과 같은 방법으로 가능한데 이 재정의 이후에 수식 번호, 표 번호, subsection 번호 등이 자신의 의도대로 잘 바뀌는지를 살펴보아야 한다.

chngcntr라는 패키지가 이 일을 좀더 쉽게 해준다. 예를 들어

```
\counterwithout{section}{chapter}
```

와 같이 함으로써 section 카운터를 chapter 카운터로부터 분리할 수 있다. 이 때는 모양만 바뀌는 것이 아니라 chapter가 바뀌어도 section 번호는 이어서 계속된다. 이와 유사한 `\numberwithin` 또는 `\numberwithout` 명령을 $\mathcal{AMS}$-math 패키지가 제공해주고 있기도 하다.

위에 기술한 모든 기능을 memoir는 모두 제공하고 있으므로 별도의 패키지를 로드할 필요가 없다.

6 Table 표 메뉴

꽤 복잡한 주제……이다. *table*은 (4.12절에서 설명한) 플로트(떠다니는 개체)로서 한 페이지 안에서만 존재한다. 즉 다음 페이지로 페이지가 나누어지지 않는 것이 원칙이다. 이 속에 들어가는 것은 대부분 `tabular` 환경인데 물론 다른 것도 얼마든지 올 수 있다. table의 폭은 그 내용물 길이에 따라 스스로 조절되는 것이 기본이다.

즉, tabular는 열과 행을 가지는 표 자체이고 table이란 이를 둘러싸서 떠다니게 해주는 조판을 위한 환경이다.

`table` 환경은 '플로트'임을 강조해둔다. 그러나 `tabular`는 그렇지 않다. label이나 캡션을 달지 않는 임의의 표를 만들려면 이 점을 유념해야 한다.

다음은 표의 일반적 형식이다.

```
\begin{table}[htbp] % placement: here, top, bottom, separate page
% \begin{small}     % sets the table font
\begin{center}      % optional
% 4-column table; alignment is left, centered, right, fixed width
\begin{tabular}{|l|c|rp{4cm}|}
\hline              % horizontal line
\textbf{Left} & \textbf{Centre} & \textbf{Right} & \textbf{4 cm}\\
\hline
row 1, col 1 & row 1, col 2 & row 1, col 3 & row 1, col 4\\
```

Left	Centre	Right	4 cm
row 1, col 1	row 1, col 2	row 1, col 3	row 1, col 4
row 2, col 1	row 2, col 2	row 2, col 3	row 2, col 4
spanning two columns		row 3, col 3	row 3, col 4
row 4, col 1	row 4, col 2	row 4, col 3	right
row 5, col 1	row 5, col 2	row 5, col 3	left
row 5, col 1	row 5, col 2	row 5, col 3	centre

표 8: A sample table.

```
\cline{1-2}         % horizontal line spanning columns 1-2
row 2, col 1 & row 2, col 2 & row 2, col 3 & row 2, col 4\\
\cline{1-2}
\multicolumn{2}{|c|}{spanning two columns} & row 3, col 3 &
row 3, col 4\\
\cline{1-3}
row 4, col 1 & row 4, col 2 & row 4, col 3 & ~ \hfill right\\
% force a space with "\ "
row 5, col 1 & row 5, col 2 & row 5, col 3 & left \hfill ~\\
row 5, col 1 & row 5, col 2 & row 5, col 3 &
~ \hfill centre \hfill ~\\
\hline
\end{tabular}
\caption{A sample table.}
% labels are used for cross references;
% for example, "see Table~\ref{tab:sampletab}"
\label{tab:sampletab}
\end{center}
% \end{small}
\end{table}
```

이 표를 조판한 결과는 표 8에서 볼 수 있다.

이따금 table의 폭이 너무 넓어서 페이지에 맞지 않을 때가 있다. 이럴 경우 rotating 패키지가 `sidewaystable`이라는 새로운 환경을 제공해준다. rotating은 이밖에도 하나의 셀 내용을 주어진 각도만큼 회전시키는 것도 가능하다. 끝으로 tabularx 패키지는 전체 테이블이 일정한 폭을 갖도록 지정할 수 있다. `X` 컬럼지시자는 필요한 만큼 컬럼의 길이를 계산해서 늘려주도록 하는 데 사용된다.

hvfloat나 floatrow 패키지들은 회전된 표를 만드는 다양한 방법을 제공한다. 또한 tabulary 패키지는 tabularx를 확장한 기능들을 제공한다. 더 확장된 표 제작 패키지는 tabu이다.

다음 보기를 보자.

```
\begin{sidewaystable}
  \begin{tabularx}{7.5cm}{|l|X|X|}
    \hline
    \textbf{normal} & \textbf{tilted} &
    \textbf{wider}\\
    \hline
    normal & \rotatebox{30}{I'm tilted!} &
    I'm wider\\
    \hline
  \end{tabularx}
\end{sidewaystable}
```

normal	tilted	wider
normal	I'm tilted!	wider

표준 tabular 환경은 페이지를 넘길 수 없다. 몇 페이지에 걸치는 긴 표를 그려야 하는 경우도 드물지 않은데, 이 제한을 극복하기 위한 패키지들이 몇 개 있어서 여러 페이지에 걸치는 표를 그릴 수 있도록 해준다. longtable, supertabular, xtab 등이 그러하다.

table 안에서 색상을 쓰려면 colortbl 패키지가 필요하다.

xcolor 패키지도 \rowcolors 명령을 제공한다.

```
1  Colour by row:\\\vskip 2mm
2  \begin{tabular}{|l|c|r|}
3    \hline \rowcolor{cyan}
4    one & two & three\\
5    \rowcolor{green}
6    one & two & three\\
7    \rowcolor{yellow}
8    one & two & three\\ \hline
9  \end{tabular}
```

Colour by row:

one	two	three
one	two	three
one	two	three

```
1  Colour by column:\\\vskip 2mm
2  \begin{tabular}
3    {|>{\columncolor{cyan}}l|
4    >{\color{red}
5    \columncolor{green}}c|
6    >{\columncolor{yellow}}r|}
7    \hline  one & two & three\\
8    one & two & three\\
9    one & two & three\\  \hline
10 \end{tabular}
```

Colour by column:

one	two	three
one	two	three
one	two	three

깔끔한 트릭 하나. LaTeX으로 표를 그리는 것이 너무 복잡하다고 생각된다면 OpenOffice Calc와 Calc2LaTeX을 사용해보라. OpenOffice는 자유 스프레드시트 프로그램이고 Calc2LaTeX은 확장모듈인데 일정한 셀 범위를 LaTeX 테이블로 변환해준다. http://www.openoffice.org/, http://calc2latex.sourceforge.net/.

가끔 편집자들은 세로 괘선이 조판상 그다지 보기 좋지 않기 때문에 가로 괘선만으로 그리라고 요구하는 경우가 있다. 만약 세로 괘선이 전혀 없다면 booktabs 패키지를 이용하여 더 세련된 표를 만드는 것이 가능하다. memoir에는 이 패키지의 기능이 이미 포함되어 있으므로 별도로 로드하지 않아도 된다.

6.1 Table/Line Spacing 표의 행간격

행의 높이는 그 안에 오는 텍스트의 높이에 따라 스스로 조절한다. 행 시작 **이전**에 간격을 추가하려면 특정 높이(height)와 0 길이를 가진 \rule로 시작하는 트릭을 쓸 수 있다. 행 **다음**에 간격을 추가하려면 \\에 추가 간격값을 선택 인자로 지정한다.

[역자 추가] 표 전체의 행간격을 일괄해서 바꾸려 할 때는 \arraystretch 값(기본값은 1)을 변경(\renewcommand)시켜준다. 단 이 값을 전역적으로 바꾸면 tabular만이 아니라 수식의 array도 영향을 받는다. 한 가지 더 추가하자면 setspace 패키지(memoir에는 이미 포함되어 있음)는 플로트 안의 내용(과 각주)을 행간격 1.0으로 조절하는 기능을 가지고 있다. 그러므로 table 안의 내용은 문서의 행간격과 별개로 기본 행간격 1.0으로 조판된다. 물론 \arraystretch로 이를 바꾸는 것이 가능하다.

```
\begin{tabular}{lll}
one & two & three\\
0.3 centimeters & \textbf{after} &
  this line\\[0.3cm]
one & two & three\\
one & two & three\\
\rule{0pt}{1.2cm}1.2 centimeters &
  \textbf{before} & this line\\
\end{tabular}
```

one	two	three
0.3 centimeters	**after**	this line
one	two	three
one	two	three
1.2 centimeters	**before**	this line

6.2 Table/Rule Width 괘선의 굵기

```
\begin{tabular}{|lll|}
\hline
%\setlength{\arrayrulewidth}{5pt}
one & two & three\\
\hline
four & five & six\\
%\setlength{\arrayrulewidth}{1pt}
\hline
\end{tabular}
```

one	two	three
four	five	six

[역자] 앞서 언급한 booktabs는 `\toprule` `\midrule` `\bottomrule` 명령을 제공하는데 이를 통하여 세로 괘선이 없는 표에서 가로 선의 굵기를 조절해준다.

6.3 Table/Aligning Numbers 숫자 정렬

테이블 안의 숫자들을 소수점 기준으로 정렬해야 하는 경우가 있다.

가장 간단한 방법은 `@` 컬럼지시자를 이용하는 것인데 셀 안에 숫자만 있을 때는 쓸만하다. 컬럼분리자 `&`가 소수점으로 대치되게 하는 트릭이다.

```
\begin{tabular}{r@{.}l}
 3&14159\\
 1&61803\\
 1&41421\\
 100&00000
\end{tabular}
```

3.14159
1.61803
1.41421
100.00000

다른 방법으로 dcolumn 패키지를 사용한다. 이 패키지는 `D` 컬럼지시자를 추가해주는데 세 개의 인자를 갖는다. LaTeX 소스와 출력에서 사용할 분리자(보통 둘 다 동일하게 '.'를 쓴다), 세 번째 것은 소수점 아래 표시할 자릿수. 세 번째 인자는 4.3과 같이 점 기준 왼쪽과 오른쪽 자릿수를 지정할 수도 있다. 이 값이 -1이면 컬럼 내용은 분리자를 기준으로 가운데정렬된다.

이 컬럼의 모든 내용은 숫자라고 간주되어 수학 모드로 조판된다. 따라서 첫 행에 헤딩을 넣고자 한다면 \mbox 안에 텍스트를 두어야 한다.

amsmath를 선언하였다면 \mbox 대신 \text를 쓸 수 있다.

```
\begin{tabular}{|D{.}{,}{4.2}|D{.}{.}{5}|D{.}{.}{-1}|}
\hline  \mbox{One} & \mbox{Two} &
\mbox{Three}\\
10.33 & 10.33 & 10.33\\
1000 & 1000 & 1000\\
5.1 & 5.1 & 5.1\\
3.14 & 3.14159 & 3.14159\\ \hline
\end{tabular}
```

One	Two	Three
10,33	10.33	10.33
1000	1000	1000
5,1	5.1	5.1
3,14	3.14159	3.14159

원문은 slashbox에 대한 내용이다. 그러나 이 패키지는 tex live에서 찾을 수 없으며 별도로 설치하지 않으면 안된다. diagbox는 slashbox에 대한 호환 명령을 제공하며 더 나은 기능을 갖추고 있으므로 이 절의 제목과 내용을 수정한다.

6.4 diagbox 이용하기

\backslashbox 명령을 이용한다.

```
\begin{tabular}{|l|l|l|}
  \hline
  \backslashbox[2cm]{Lesson}{Date} &
  Monday & Tuesday\\
  \hline
  Stratigraphy & room A & room A\\
  Chemistry & room B & Lab $\alpha$\\
  Physics & room C & Lab $\beta$\\
  \hline
\end{tabular}
```

Lesson \ Date	Monday	Tuesday
Stratigraphy	room A	room A
Chemistry	room B	Lab α
Physics	room C	Lab β

6.5 그밖에 재미난 것

이 절은 역자가 임의로 추가한 것임

테이블에 관련해서는 재미있는 것이 많이 있다. 그만큼 문제점도 많은 셈이다. 여기서는 tabularcalc라는 것을 소개한다. 자세한 것은 패키지 문서를 읽으면 되고 샘플을 보이면 다음과 같다.

```
\htablecalc[3]{$x$}{x=-4,-2,0,2.25,7}
{$2x-3$}{2*x-3}
{$x^2$}{x*x}
{$\sqrt{x^2+1}$}{round(root(2,x*x+1),3)}
```

x	-4	-2	0	2.25	7
$2x-3$	-11	-7	-3	1.5	11
x^2	16	4	0	$5.062{,}5$	49
$\sqrt{x^2+1}$	4.123	2.236	1	2.462	7.071

이 테이블에서 보다시피, x 값만 소스에 지정하면 결과는 TEX이 계산해주고 있다.

xcolor는 \rowcolors 명령을 제공한다. colortbl에도 비슷한 명령이 있지만 이 쪽이 조금 더 재미있다. \rowcolors를 쓰려면 xcolor에 [table] 옵션을 지정해야 한다.

KTUG의 Faq 문서 'Tabular 환경'은 표 문제에 관한 거의 모든 해결책을 모아 둔 곳이다. 꼭 한 번 읽어볼 필요가 있다. 또한 tabu 패키지도 참고하라.

6.6 데이터를 표로 들여오기

데이터 파일은 우리가 일상적으로 활용하는 것이다. 대부분의 데이터 파일은 컬럼별로 숫자가 적혀 있는 아스키 텍스트 파일이다. 그리고 스프레드시트를 사용하는 사람도 많다. 거의 모든 스프레드시트 프로그램은 아스키 형식의 데이터 파일(.csv)로 내보낼 수 있다. 이 파일에서 각 구획은 ';' 문자로 구분되는 것이 일반적이다.

데이터 파일을 LaTeX 표로 변환하는 것은 지루한 일일 수밖에 없다. 다음 스크립트는 Unix에서 데이터 파일을 임의의 컬럼수를 가진 테이블로 변환하는 것이다. .csv 파일에도 잘 동작한다.

```
#!/bin/sh

# dat2tex.sh: converts tabular data to a tabular environment

if [ $# != 1 ]; then
  echo "Usage: $0 <datafile>"
  exit 1
fi

# is this a csv file?
grep ";" $1 > /dev/null
if [ $? = 0 ]; then
  AWK="awk -F;"
else
  AWK=awk
fi

# ok awk, make my day
$AWK '{if (1 == FNR) { \
        printf "\\begin{tabular}{"; \
        for (i = 1; i <= NF; i++) {printf "l"}; \
        printf "}\n"
      }
      for (i = 1; i < NF; i++) \
        {printf $i" & "} printf $NF"\\\\ \n"} \
      END {printf "\\end{tabular}\n"}' $1

# end of dat2tex
```

7 Tools 도구 메뉴

7.1 Tools/Mail Merges 메일 머지

이 유용하고 시간을 절약하게 해주는 도구는 textmerg 패키지로 구현된다. 어떤 문서가 있는데 거기에 표시되는 사람의 성과 이름, 호칭만을 바꾸어서 여러 장을 만들어야 하는 경우를 생각해보자. 그밖의 텍스트는 모두 똑같다.

세 개의 *field*를 정의한다. 각각 \Name, \Surname, \Title이라 하고 이것이 문서의 변하는 부분이 된다. 각각의 값은 외부 파일 data.dat로부터 가져오기로 하자.

```
\documentclass{article}
\usepackage{textmerg}
\begin{document}
```

```
% let's declare the variable fields:
% \Void is for empty lines
\Fields{\Name\Surname\Title-\Void}
\Merge{data.dat}{%
Dear \Title{} \Surname,\\
may I call you \Name?\\
Yours,\\
\hspace{3cm}Guido\clearpage}
\end{document}
```

네 번째 필드 \Void는 실제로 꼭 있어야 하는 것은 아니다. 그 앞에 마이너스 부호를 붙인 것은 데이터 파일에서 비어 있을 수 있다는 뜻이다. 이것을 둠으로써 레코드 사이를 빈 줄로 분리하려는 것이다.

data.dat 파일은 다음과 같이 작성한다.

```
Guido
Gonzato
Dr.

Francesco
Mulargia
Prof.

Marie
Curie
Mme
```

이것으로 준비는 끝났다. 출력되는 결과를 보면 불러온 텍스트가 포함되어 각 수신자마다 한 페이지씩 문서가 만들어진다.

7.2 Tools/Labels 이름표 만들기

메일 머지가 쉬웠다면 라벨 만들기는 훨씬 간단하다. 3×8 라벨용지에 20개의 라벨을 만들어야 하는 경우를 생각해보자. 사용할 패키지는, 짐작하겠지만 labels라는 것이다. 이 예에서 10개의 보통 라벨과 10개의 박스 라벨을 만들려고 한다.

```
\documentclass[a4paper,12pt]{article}
\usepackage{labels}
\LabelCols=3      % n. of columns of labels
\LabelRows=8      % n. of rows of labels
\LeftBorder=8mm   % borders of each label
\RightBorder=8mm
\TopBorder=5mm
\BottomBorder=5mm
\LabelGridtrue      % show the grid
\numberoflabels=10  % number of labels of each type to print
% the text of the label is specified by
% the \addresslabel[]{} macro:
\begin{document}
  \addresslabel[\large] % optional arguments
  {\textbf{Guido Gonzato}, Ph.D.\\
  \textsl{Linux system manager}}
```

```
  % now on to the boxed labels
  \boxedaddresslabel[\fboxsep=4mm\fboxrule=1mm]
  {\textbf{Guido Gonzato}, Ph.D.\\
  \textsl{Linux system manager}}
\end{document}
```

서로 다른 주소를 포함한 라벨을 만들려면 외부 파일을 이용할 수도 있고 메인 파일에 주소를 적어넣을 수도 있다.

```
\documentclass[a4paper,12pt]{article}
\usepackage{labels}
\LabelCols=3
\LabelRows=8
\LeftBorder=3mm
\RightBorder=3mm
\TopBorder=8mm
\BottomBorder=8mm
\LabelGridtrue
\begin{document}
% use either this environment:
\begin{labels}
  1$^{st}$ name
  1$^{st}$ address
  1$^{st}$ city, state, zipcode

  2$^{nd}$ name
  2$^{nd}$ address
  2$^{nd}$ city, state, zipcode

  3$^{rd}$ name
  3$^{rd}$ address
  3$^{rd}$ city, state, zipcode
\end{labels}
% or an external file containing exactly the same text:
% \labelfile{addresses.dat}
\end{document}
```

textmerg와 labels를 함께 쓰는 것도 가능하다. 한번 해보시라!

7.3 Tools/Default Language 다국어

LaTeX의 기본 언어는 영어이다. 다른 언어도 지원한다. '언어 지원'이란 말은 예를 들면 'Chapter'나 'Index'와 같은 용어의 번역, 정확한 하이픈, 'ç'나 'é' 같은 문자를 키보드로부터 직접 입력할 수 있는 기능 등을 의미하는 것이다. (이 글자들은 보통 `\c c`나 `\'e`와 같은 방식으로 입력한다.)

XeTeX이나 LuaTeX과 같은 유니코드 텍 시스템이 도입됨으로써 언어 지원은 이전과 전혀 다른 양상이 되었다고 할 수 있다. XeTeX에서는 babel 대신 polyglossia 패키지를 써서 이전에 babel이 하던 역할을 대신한다. 한편 한국어 지원은 koTeX이 담당하고 있다.

LaTeX 배포판에는 `language.dat`라는 파일이 들어 있다(보통 `$TEXMF/tex/generic/config/language.dat`). 이 파일에는 언어의 목록이 포함되어 있는데 이를 수정함으로써 원하는 언어의 하이픈 패턴을 선택할 수 있다.

만약 영어 사용자가 아니라면 babel 패키지를 다음과 같이 사용하면 된다.

```
\usepackage[italian,english]{babel}
```

☞ babel은 몇몇 문자의 동작을 해당 언어에 맞추어 변경시킨다. 글자가 사라지는 등 이상한 문제를 경험하면 나타나지 않는 글자를 \charXX 문법으로 삽입하면 될 것이다.

또한, 표준 ASCII 문자[3]가 아닌 부호붙은 글자들을 입력하려면 isolatin1 패키지가 포함되어야 한다. 그러나 이것은 권장하지 않는다. 왜냐하면 그것이 소스 파일의 가독성과 이식성을 감소시키기 때문이다. 계속 TEX 방식을 사용하는 편이 낫다.

XƎTEX을 주로 사용하는 한국어 사용자가 기억해야 할 것이 있다. 이 글에서 언급하는 유럽어 입력에 관계된 inputenc 패키지는 XƎTEX에서 사용할 수 없다. 그러므로 예전 레거시 텍에서 작업하던 파일을 XƎTEX으로 처리하고자 한다면 이 패키지 부분을 주석처리해야 한다. 그리고 babel 패키지도 XƎTEX과는 맞지 않으므로 만약 이 패키지가 사용된 다국어 문서라면 적절하게 (polyglossia를 이용하도록) 수정해야 한다.

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

한 글자를 입력하기 위해서 서너 번의 키조작을 해야 하는 것이 불만이라면 에디터 설정을 바꾸어서 이를 해결할 수 있을 것이다. 예를 들어 내가 쓰는 jed에서 'é'를 타이프하면 에디터가 이를 \'e로 바꾸어 입력하도록 설정해두었다. 이를 위한 나의 .jedrc이다.

```
define latex_mode_hook ()
{
  set_abbrev_mode (1);
  if ( () = abbrev_table_p ("LaTeX") )
    use_abbrev_table ("LaTeX");
#ifdef WIN32
  % prevent clash with movement keys
  undefinekey ("àà", "LaTeX-Mode");
  definekey (" \\`a", "àà", "LaTeX-Mode");
#else
  local_setkey (" \\`a",     "à");
#endif
  local_setkey (" \\'e",     "é");
  local_setkey (" \\`e",     "è");
  local_setkey (" \\`\\i{}", "ì");
  local_setkey (" \\`o",     "ò");
  local_setkey (" \\`u",     "ù");
}
```

자신의 에디터에 알맞은 방법은 에디터 도움말을 통해 찾아보라.

7.4 Tools/Hyphenation 하이픈 설정

LATEX이 대체로 하이픈 처리에 있어서 훌륭하지만 가끔 수작업으로 개입하는 것이 더 나은 결과를 가져올 때가 있다. 수동 하이픈은 \-를 단어 중간 하이픈이 필요한 곳에 적어넣으면 된다. 더 나은 방법으로 하이프네이션 규칙을 선언하는 방법도 있다.

하이픈 지정한 단어와 단어 사이에 콤마가 없음에 유의하라.

```
\hyphenation{ge-o-phy-sics ge-o-lo-gy earth}
```

위의 예에서 'earth'라는 단어는 하이픈 처리하지 말도록 LATEX에게 알려주고 있는 것이다. 특정 단어가 잘라지지 않게 하는 다른 방법으로 단어 전체를 \mbox 안에 넣어주는 방법도 있다.

[3] 컴퓨터 용어로서 '표준 ASCII 문자'라 함은 code 32(space)에서 126(tilde)까지의 글자들을 말한다.

```
Do not hyphen \mbox{internationalisation}, please. I'm a masochistic.
```

7.5 Tools/Spell Check 철자검사

LaTeX 자체는 철자검사와 아무 관련이 없다. 이것은 `ispell`, `aspell` 등 외부 프로그램을 이용해서 해야 한다. Unix에서 `ispell`을 다음과 같이 사용한다.

한국어 철자 검사는 명령행 인터페이스를 가진 것이 별로 없다. 아무튼 철자 검사가 에디터와 연동되는 것이 가장 바람직할 것이다. 다행히 오픈소스 한국어 철자검사기가 hunspell 사전으로 나와 있기 때문에 이를 이용하는 방법이 현재 개발되어 가고 있다. 예컨대 TeXworks 에디터는 hunspell 라이브러리를 채용하고 있으므로 부분적으로 한국어 철자검사가 가능해졌다.

```
shell> ispell -t mydocument.tex
```

`-t` 옵션은 `ispell`에게 TeX 및 LaTeX 명령을 무시하라고 알려주는 것이다. 주언어가 영어가 아니라면 적절한 사전을 `-d` 옵션으로 지정해주면 된다.

```
shell> ispell -d italiano -t mydocument.tex
```

8 Help 메뉴

LaTeX에 대한 도움말을 얻는 데는 많은 방법이 있다. 온라인은 물론이고 오프라인 서적도 많다. 가장 좋은 출발점은 CTAN 사이트일 것이다. http://www.ctan.org/tex-archive/info/.

- `info latex` (Unix systems) 각종 명령, 개념에 대한 소략하지만 완전한 요약본을 보여주는 명령이다.
- http://www.giss.nasa.gov/latex/은 종합적인 온라인 참고문서이다. 유용한 링크가 많다.
- http://www.ctan.org/tex-archive/info/LatexHelpBook/은 LaTeX에 대한 훌륭한 도움말 시스템이다. Windows와 잘 통합되어 있다.
- news:comp.text.tex 뉴스그룹을 잊지 말자. 매우 유용한 도움말을 얻을 수 있다.

그리고 역자는 KTUG을 이 목록에 추가하지 않을 수 없다.

2015년 현재, 대부분의 GNU/Linux 배포판은 `TeX Live`를 포함하고 있다. 많은 문서가 함께 제공되므로 설치본에서 문서를 찾아 읽을 수 있으면 좋다. `/usr/share/doc/texlive-doc/` 아래에서 찾을 수 있을 것이다.

이와 더불어, `tex live`를 설치하였다면 `texdoc`이라는 유틸리티를 활용할 수 있다.

9 마지막 말

 이 가이드가 유용하기를 진심으로 바란다. 제안과 비평이 있다면 저자에게 연락해주기 바란다.

번역본의 라이센스도 동일하다. 번역에 관련된 문제는 역자에게 연락해주기 바란다.

부록 A 문서 샘플

`article` 클래스 템플릿은 2.1절에서 보였다. 아래는 추가적인 샘플 문서들이다.

```
\documentclass[twoside,11pt]{book}
\begin{document}
\frontmatter
\begin{titlepage}
\title{The Book of Mine}
\end{titlepage}
\author{John B. Smith}
\maketitle
\tableofcontents
\mainmatter
\part{The Beginning}
\chapter{Introduction}
\section{Let's Start}
The book starts here.
\part{The End}
\backmatter
Thank you for reading this book.
\end{document}
```

그림 A.1: Book template.

```
\documentclass[twoside,12pt]{report}
% tables and figures at the end:
\usepackage{endfloat}
\begin{document}
\title{Final Report}
\author{John B. Smith}
\date{London, \today}
\maketitle
\begin{abstract}
This is the final report.
\end{abstract}
\tableofcontents
\listoftables
\listoffigures
\part{Start}
\chapter{Begin}
\section{Introduction}
The report starts here.
\end{document}
```

그림 A.2: Report template.

```
\documentclass[12pt]{letter}
\begin{document}
\address{My address}
\signature{Guido}
\begin{letter}{John's address}
\opening{Dear John,}
Thank you for being my friend.
\closing{Hope to see you soon,}
\ps{P.S. Say hello to granny!}
\encl{My son's photographs!}
\end{letter}
\end{document}
```

그림 A.3: Letter template.

```
\documentclass[a4paper]{article}
\usepackage{type1cm}
\usepackage{times}
\usepackage{color}
\usepackage{rotating}
\pagestyle{empty}
\begin{document}
\begin{sidewaysfigure}
  \fontsize{2.5cm}{2.5cm}\selectfont
  \centerline{\textcolor{blue}{\textbf{Please:}}}
  \vskip 1cm
  \fontsize{4cm}{3cm}\selectfont
  \centerline{\textcolor{red}{DO NOT}}
  \centerline{\textcolor{red}{SMOKE}}
  \centerline{\textcolor{red}{HERE!}}
  \vskip 1cm
  \fontsize{2cm}{2cm}\selectfont
  \centerline{\textcolor{magenta}{If you do,}}
  \centerline{\textcolor{magenta}{you'll be \emph{deboned!}}}
\end{sidewaysfigure}
\end{document}
```

그림 A.4: 안내문 작성 예제.

```
\documentclass{article}
\usepackage[absolute,showboxes]{textpos}
\usepackage{color}
\usepackage{framed}
\usepackage{graphicx}
\setlength{\TPHorizModule}{10mm} % standard unit of length
\setlength{\TPVertModule}{\TPHorizModule}
\setlength{\TPboxrulesize}{1pt}  % box line width
% start everything near the top-left corner
\textblockorigin{0mm}{0mm}

\begin{document}
\setlength{\parindent}{0pt}
\definecolor{shadecolor}{rgb}{0.9,1,1}
\begin{textblock}{5}(0,0)
% this block is 5 modules wide; height is
% automatically determined
\begin{center}
  \begin{minipage}[c]{0.8 \linewidth}
  \begin{shaded}
  This block is placed with its top left corner at the `origin'
  on the page, which has been set to (0mm,0mm). The internal
  margin and the shading are provided by the \texttt{minipage}
  and \texttt{shaded} environments.
  \end{shaded}
  \end{minipage}
\end{center}
\end{textblock}
\begin{textblock}{6}(10,1)
  \includegraphics[width=6cm,angle=-90]{gnuplot.ps}
  This picture is at (10,1). Note that rotating it
  by -90 makes it overflow the margin.
\end{textblock}
\begin{textblock}{5}[0.5,0.5](2.5,8)
This block is at position (2.5,8), but because the optional
argument [0.5,0.5] has been given, it is the centre of the block
which is located at that point, rather than the top-left corner.
\end{textblock}
\begin{textblock}{3,4}(6,4)
The dimensions of this block are 3$\times$4 cm.
Its origin is position (6,4) on the page. Note that the text
overflows the margin in some cases; you'll want to
use the \texttt{minipage} environment to prevent that.
\end{textblock}
\end{document}
```

그림 A.5: 포스터 작성 예제.

부록 B XeTeX에서 본문 폰트 설정하기

이 절은 XeTeX의 폰트 선택 명령을 간략하게 기술한다. 수식 글꼴에 대해서는 다루지 않는다.

B.1 NFSS

LaTeX 2ε의 글꼴 선택 명령을 이해하기 위해서는 다음을 알아야 한다.

`encoding` 이른바 "폰트 인코딩"이라는 것이다. 레거시 텍에서 매우 중요하게 취급하는 것으로 ASCII를 대상으로 하는 OT1 인코딩과 유럽 문자까지 포함하는 T1 인코딩이 특히 중요하다. 그러나 XeTeX에서는 크게 문제삼지 않아도 상관없다.

`family` 글꼴 가족이라 불리는 것. rm, sf, tt 세 가지 패밀리가 주로 쓰인다. rm은 세리프체, sf는 산세리프체, tt는 모노스페이스 폰트에 대응한다. 우리 식으로 말하자면 각각 바탕체, 돋움체, 타자체 정도이다. 즉, '서체군'을 가리키는 말이다.

`series` mdseries, bfseries가 있고, 각각 보통체(medium), 굵은체(boldface)를 의미한다.

`shape` upshape, itshape, scshape, slshape가 있다. 각각 바로선 모양(upright), 이탤릭 모양(italic), 작은대문자 모양(smallcapital shape), 기울인 모양(slanted)을 의미한다. 이탤릭과 기울임체는 다르다. 예를 들면 *italic*, *slanted*.

`size` 글꼴의 크기이다. 이것은 각각 별개의 명령으로 주어지는데, `\tiny`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge`가 있다. 글꼴 크기 지정 명령은 다른 속성과 독립적으로 동작한다.

이제 5.2절에서 설명한 폰트 선택 명령의 의미를 알 수 있을 것이다.

B.2 fontspec

레거시 텍에서는 위의 명령만으로 문제가 없었다. 그러나 트루타입/오픈타입 폰트를 활용하려면 `fontspec` 패키지가 필요하다. 이 패키지는 특히 다음 세 명령을 제공하는데,

- `\setmainfont`
- `\setsansfont`
- `\setmonofont`

각각 rm, sf, tt 패밀리 글꼴을 지정하는 것이다. 예를 들어

```
\setmainfont{Times New Roman}
```

이 명령은 Times New Roman 서체를 rm 패밀리에 대응시키라는 말이 된다. 이제 문서의 본문의 기본 글꼴이 이 서체가 될 것이며, `\rmfamily` 명령도 동작한다.

한편, series나 shape는 어떻게 구현하는가? 이들은 위의 세 명령에 옵션 인자로 지시해야 한다. 예를 들어 Times New Roman 서체의 경우는 Boldface를 별도로 지정하지 않아도

글꼴 가족 정보로부터 이를 잘 읽어온다. 그런데 다음과 같은 경우, 즉 "폰트 이름으로" 호출하지 않고 "파일 이름으로" 불러온 경우에는 bold에 해당하는 글꼴 이름을 적어주어야 한다.

```
\setmainfont{timesnewroman.ttf}[BoldFont={timesnewroman-bold.ttf}]
```

`BoldFont, ItalicFont, SmallCapsFont, BoldItalicFont` 옵션으로 각각에 해당하는 글꼴을 전부 적어주어야 하는 극단적인 경우도 있을 것이다. 그러나 대부분의 잘 설계된 폰트들은 이 글꼴가족군을 스스로 찾아서 식자한다. 단, 이렇게 하려면 글꼴을 "글꼴 이름으로" 지정해야 한다.

글꼴 이름으로 호출한다는 것은 무슨 의미인가? 예를 들어 "`times.ttf`"와 "`timesbd.ttf`", "`timesit.ttf`", "`timesbdit.ttf`"라는 파일들이 시스템에 등록되면 "Times"라는 이름을 갖는다고 하자. 이 때 위의 각각의 파일 이름과 다른 폰트를 가리키는 이름(family name)이 존재한다. 특정 폰트의 폰트 이름을 알려면 `otfinfo -i` 명령으로 확인할 수 있다. 또는 폰트 유틸리티를 활용할 수도 있다.

파일이름으로 불러올 때는 확장자를 붙이면 된다.

`\setmainfont{times.ttf}`.

위와 같은 방식으로 글꼴의 스펙을 지정한 후에는, 익숙한 NFSS 명령을 그대로 쓰면 된다. 가끔 NFSS 명령이 아니라 부분적으로 특정 폰트만 쓰고 싶을 때 `\fontspec` 명령을 그대로 쓸 수도 있으나, 일반적으로 권장할 방법은 아니다. 즉, 본문에서는 NFSS 명령만 사용하라. 그리고 어떤 폰트를 각 패밀리와 시리즈, 셰이프에 할당할 것인지는 fontspec 패키지로 설정해두어라.

B.3 ko.TEX의 글꼴 설정 명령

ko.TEX은 글꼴을 세 영역으로 구분한다. 라틴문자, 한글, 한자이다. 그리고 라틴문자군에만 fontspec의 글꼴 설정을 그대로 활용한다. 한글과 한자에 대해서 각각 별도의 fontspec-류 명령이 존재한다.

```
\setmainhangulfont    \setmainhanjafont
\setsanshangulfont    \setsanshanjafont
\setmonohangulfont    \setmonohanjafont
```

사용법은 위의 fontspec 명령과 거의 같고 다만 한글/한자를 위한 속성 옵션 몇 가지가 추가되어 있다. 예를 들면 한글 자간을 설정하는 InterHangul 같은 것이 그러하다. 이 명령의 사용법은 해당 패키지 문서를 참고하여야 한다.

```
\setmainhangulfont{HCR Batang LVT}[ItalicFont={HCR Dotum LVT}]
```

위의 문장은 한글 글꼴을 설정하는 하나의 예시이다. 이 선언은 한글(과 한자)에만 적용되고 라틴문자에는 적용되지 않는다.

그러나 본문에서는 여전히 NFSS 명령으로 이를 적용할 수 있다. 예컨대 `\textrm`은 한글에서는 HCR Batang LVT 글꼴로 식자되게 한다.