# KC2008Plus 설치 후 첫 예제

KTUG Collection Team

2011년 10월 7일

이 부분은 처음 TeX을 접하시는 분들을 위한 간단한 예제입니다. KC2008Plus가 설치되었습니다.

KC2008Plus는 ko.TeX Live에서 한글을 사용하기 위한 도구를 통합한 결과물입니다.

여러분은 지금 KC2008Plus에 포함된 Notepad++을 통해 이 문서의 소스를 보고 계십니다.

이 문서의 pdf파일을 얻으려면 [Ctrl-Alt-F6]을 눌러 보십시오. 이것은 XeLaTeX을 실행하는 것입니다. [Ctrl-Alt-F5] 키를 누르면 한번 컴파일하여 결과를 보여줍니다. 또는 메뉴의 "플러그인-KCmenu"을 열어보시기 바랍니다. KCmenu는 plugin–KCmenu–Call KCmenu로도 열 수 있습니다.

다른 문의사항은 http://www.ktug.or.kr/의 'TnXTeX 마당'을 이용해 주십시오.

감사합니다. 즐텍 Happy TEX'ing.

# KC2008Plus 설치 후 첫 예제

KTUG Collection Team

2011년 10월 7일

**요 약**

KC2008Plus가 설치되었습니다. KC2008Plus는 ko.TeX Live 2011이 설치되어야 잘 동작합니다. 여러분은 지금 KC2008Plus에 포함된 Notepad++을 통해 이 문서의 소스를 보고 계십니다.

- XeLaTeX으로 컴파일하여[1] `pdf`를 볼 수 있습니다. XeLaTeX컴파일을 위해서는 Ctrl-Alt-F6 또는 Ctrl-Alt-F5를 누릅니다. OpenType 또는 TrueType 폰트를 사용합니다. 이 예제 문서에서 라틴 폰트는 로만 계열로 Palatino 계통인 TeX Gyre Pagella, 모노스페이스 계열로 Consolas를 사용하도록 했습니다. 한글 폰트는 명조 계열로 함초롬 바탕 LVT, 고딕 계열로 함초롬 돋움 LVT를 씁니다.
- 예전 방식으로 `pdf` 파일을 얻으려면 F6을 눌러 보십시오. 이것은 pdfLaTeX을 실행하는 것입니다. F5 키를 누르면 한번 컴파일하여 결과를 보여줍니다. 또는 메뉴의 "플러그인-KCmenu"을 열어보시기 바랍니다. KCmenu는 plugin–KCmenu–Call KCmenu로도 열 수 있습니다.
- Alt-F5 또는 Alt-F7을 누르면 dvipdfm$x$를 거쳐 `pdf`를 볼 수 있습니다. `dvi` 출력물을 반드시 얻어야 하는 경우라면(권장하지 않습니다) Alt-F6을 누르면 됩니다. `dvi` 파일의 화면 출력이나 변환은 명령행을 통해서 하십시오.
- pstricks를 이용한다든지 `ps`를 거쳐 `pdf`를 얻어야 할 때가 있습니다. 그런 경우, Ctrl-F6과 Ctrl-F7 단축키를 이용하십시오.
- LuaTeX으로 컴파일하여[2] `pdf`를 볼 수도 있습니다. LuaLaTeX으로 컴파일하려면 Alt-A를 눌러서 명령창을 연 다음 명령을 내리십시오.

  ```
  > lualatex first
  ```

- 다른 문의사항은 `http://www.ktug.or.kr/`의 'TnXTeX 마당'을 이용해 주십시오.

감사합니다. 즐텍 Happy TeX'ing.

[1] XeTeX은 Jonathan Kew가 만든 오픈타입/트루타입을 이용하는 유니코드 텍 시스템입니다. XeLaTeX은 XeTeX의 LaTeX 판입니다. 이 문서는 XeLaTeX으로 조판되어 있습니다.

[2] LuaTeX은 Hans Hagen, Harmut Henkel, Taco Hoekwater가 만들고 있는 lua 스크립트 언어를 내장한 pdfTeX의 확장 시스템입니다. LuaLaTeX은 LuaTeX의 LaTeX판입니다.

# 1 돌다리 두드리기

## 1.1 최종 출력물 얻기

LaTeX 컴파일의 최종 출력물은 `pdf` 파일입니다. 전에는 `dvi`가 최종 출력물일 때도 있었지만 이제는 `pdf`가 표준 출력 형식이 되었습니다. 아마 `dvi` 포맷에 대해 들어보지 못한 분도 있으리라 생각합니다. 그러므로 특별한 경우를 제외하고는 최종 출력물이 당연히 `pdf`라고 보겠습니다. 우리가 컴파일하고자 하는 `tex` 소스의 파일 이름을 `foo.tex`이라 하지요. 간단하게 다음과 같이 하면 됩니다.

```
> xelatex foo
```

얼마 전까지만 해도 pdfLaTeX이 일반적으로 쓰이던 컴파일 명령이었습니다.[3] 그러나 한글 문서에 관한 한 현재 시점에서 표준이라고 볼 수 있는 것은 xeLaTeX입니다. 이것은 예전의 latex 명령을 실행한 것과 똑같다고 보시면 됩니다.[4]

여기에 옵션을 줄 수 있습니다. 예를 들면 pdf 거꾸로찾기 (inverse search) 를 위하여

```
> xelatex -synctex=1 foo
```

또는 중간에 오류가 있더라도 일단 끝까지 컴파일하라고

```
> xelatex -synctex=1 -interaction=nonstopmode foo
```

와 같이 하는 것이 가능합니다. 이런 세세한 컴파일 옵션은 대부분 편집기에서 자동으로 붙여줍니다.

## 1.2 그림 삽입

LaTeX 문서에 그림을 넣는 방법은 다양하게 존재했고 많은 발전을 거듭했습니다. 엄밀히 말하면 LaTeX은 그림 포맷에 대해 모릅니다. 단지 수 많은 텍스트와 마찬가지로 그림도 가로와 세로의 길이를 지닌 상자 (box) 로 인식할 뿐입니다. `\special` 명령을 이용하여 넣는 방법부터 psfig, epsfig처럼 `eps`를 겨냥한 오래된 패키지도 있었지만, 요즘은 graphicx 패키지 하나만을 얹어도 일단 그림을 넣을 수 있습니다.

그러면 어떤 형식의 그림 파일을 넣을까요? 사용자가 얻으려 하는 최종 출력물이 `pdf`이고 컴파일에 pdfLaTeX이나 XeLaTeX을 쓴다면 `pdf`와 `jpg`, `png`를 삽입할 수 있습니다. TeX Live 2011 이후버전에서는 `eps`도 무리없이 삽입합니다.

그러면 그림을 하나 넣어볼까요? `sample.pdf`입니다. (다음쪽의 그림 1 참조) 그림 이름에 확장자를 쓰지 않은 것을 주의해서 보십시오.

```
\begin{figure}
\hangcaption
```

[3] 지금도 라틴문자로 된 문서의 경우 pdfLaTeX이 널리 쓰입니다.

[4] XeLaTeX 이외에 차세대 텍 엔진으로서 LuaLaTeX이 주목받고 있습니다. 언젠가는 이것이 주류가 되겠지만 아직은 테스트 단계라고 보셔도 될 것 같습니다.

그림 1: Garamond 폰트의 long s-i 리거처 (출처 : WIKIPEDIA)

```
\captionnamefont{\small\sffamily}
\captiontitlefont{\small}
 \centering{%
\includegraphics[width=.75\textwidth]{sample}}
\caption{ Garamond 폰트의 long s-i 리거처
(출처: \protect\href{http://en.wikipedia.org/wiki/Typographical_ligature}
{\textsc{WikipediA}})}
\label{fig:ligature}
\end{figure}
```

* * *

그림 넣는 방법이 예전과 조금 달라져서 당황스러운 경우를 겪을 수 있습니다. 여기 몇 가지 관련된 사항을 적어두어서 참고하시도록 하겠습니다.

dvips와 eps 이 안내글에 DVIPS에 대한 언급이 없는 것을 놀라워하시는 분도 계시리라 봅니다. DVIPS는 다음과 같은 이유 때문에 일반적인 문서 작성에 적당하지 않습니다. 즉, DVIPS를 이용하지 않는 더 좋은 대안이 많습니다.

- 트루타입 글꼴을 처리하지 못합니다.
- eps 이외의 그림을 처리하지 못합니다.
- 경험상 eps 포맷의 그림은 문제를 일으키는 경우가 많습니다.
- 그밖에 사소한 불일치(초보자가 당황할 수 있는)를 겪을 가능성이 많습니다.
- pdf를 얻기까지 너무 많은 단계를 거칩니다.

그러나 DVIPS를 반드시 사용해야 하는 경우도 있습니다. pstricks를 쓸 때가 그렇습니다. 그러므로, 자신이 작성하는 문서의 target이 DVIPS를 위한 것인지 아닌지를 명백히 인식하고 그림을 준비해야 하는 것입니다.

DVIPDFM$x$ 한때 LATEX과 DVIPDFMX를 이용하여 문서를 컴파일하는 방법이 있었습니다. 이 때는 다음과 같이 해주어야 했습니다. 먼저, graphicx 패키지에 [dvipdfmx] 옵션을 주어야 합니다.

```
\usepackage[dvipdfmx]{graphicx}
```

그리고 LATEX 실행시에 -shell-escape 옵션을 주어야 합니다.

```
> latex -shell-escape foo
```

실제 문서에서는 \includegraphics로 위와 같이 그림을 넣었습니다. 이 절차는 바운딩박스를 제대로 얻기 위한 것이었는데 이에 대해서 항목을 달리해서 조금 설명하겠습니다.

바운딩박스 예전에는 (eps를 제외한) 모든 그래픽 파일의 bb 또는 xbb를 얻어놔야 했습니다. jpg, png를 이용하려면 해당 그림의 bb를 만들어 둔 뒤에 컴파일하면 됩니다. bb를 얻는 유틸리티로는 가장 널리 알려진 EBB를 비롯, XBB 등이 있습니다.

EXTRACTBB EXTRACTBB 유틸리티는 DVIPDFM$x$ 프로젝트에 들어있는 것으로 EBB의 몇 가지 문제점을 수정하여 확장한 것입니다. sample.jpg에 대해 EXTRACTBB를 실행하면 sample.xbb가 생깁니다. (png, pdf에 대해서도 마찬가지) EXTRACTBB를 다음과 같이 실행하면 sample.bb가 생깁니다. 이것은 EBB를 실행한 것과 같습니다.

```
> extractbb -x sample.jpg
...
\usepackage{graphicx}
...
\includegraphics{sample}
```

.xbb와 .bb XBB가 만들어진 목적 중의 하나가 DVIPDFM의 그림 처리 방법이 pdfTEX의 결과와 다른 경우가 있었다는 점을 해결하기 위한 것이었습니다. 새로 생긴 확장자 `.xbb`는 pdfTEX과 같은 결과를 얻기 위해 계산된 바운딩 박스 정보를 담습니다. 반면, 기존의 DVIPDFM과 같은 방식의 바운딩 박스를 얻으려면 EXTRACTBB를 EBB 호환되게 실행하면 되는데, 이렇게 해서 얻어지는 바운딩 박스 파일은 확장자가 `.bb`입니다. `.bb`를 얻으려면 EXTRACTBB에 `-m` 옵션을 주고 실행합니다. EXTRACTBB의 기본값은 `.xbb`입니다. 과거에 만들어진 문서의 호환성을 위해서가 아니라면 특별히 EBB를 실행하여 `.bb`를 만들어 써야 할 필요는 없을 것입니다.

pdflatex과 eps PDFTEX 엔진은 그 자체로는 `eps` 파일을 잘 처리하지 못하였습니다. 그래서 `\usepackage{epstopdf}`를 문서에서 지시하여 컴파일 과정에 임시로 `pdf`로 변환한 다음 이것을 문서에 포함하도록 하는 방법이 널리 쓰였습니다. 그러나 최근에는 이것도 아무런 지시 없이 자동으로 처리하도록 바뀌었습니다.

그러나… 모두 잊읍시다. `eps` 그림이 애를 먹이는 것도, 그림을 넣기 위해 앞에서 했던 `bb` 또는 `xbb`를 얻는 복잡한 과정도 모두 잊읍시다. 그림은 `pdf`나 `png`로 준비하면 됩니다. 그림 삽입은 `\includegraphic`면 됩니다.

## 2 몇 가지 예제

### 2.1 수식

이것은 수식 예제입니다. $ax^2 + bx + c = 0$ $(a \neq 0)$의 일반해를 구하는 공식은 근의 공식입니다.

```
\[x=\frac{-b \pm \sqrt{b^2-4ac}}{2a}\]
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

행렬은 본문에서 $(\begin{smallmatrix}1&2\\3&4\end{smallmatrix})$와 같이 식자됩니다. 별행 수식(display math)에서는 다음과 같습니다.

```
\[\bordermatrix{%
&    & i^{\text{th}} &  \cr
& A & B & C \cr
 j_{\text{th}} & d & e & f \cr
& 1 & 2 & 3 \cr}
 \]
```

$$\bordermatrix{ & & i^{\text{th}} & \cr & A & B & C \cr j_{\text{th}} & d & e & f \cr & 1 & 2 & 3 \cr}$$

여러 줄에 걸친 수식을 큰 괄호로 묶을 때 줄 바꿈이 일어나면 \left와 \right의 짝이 안 맞을 수 있습니다. 이럴 때 \big, \Big, \bigg, \Bigg 등의 명령이 유용합니다.

```
\tabson
\begin{align*}
B(r,\phi,\lambda) = & \,\dfrac {\mu}{r} \Bigg [\sum_{n =2}^{\infty}
\Bigg (\left (\dfrac {R_e}{r} \right)^n J_n P_n(s\phi) \\
& +\sum_{m =1}^n \left (\dfrac {R_e}{r} \right )^n (C_{nm}\cos m\lambda
+S_{nm}\sin m\lambda )P_{nm}(s\phi) \Bigg)\Bigg]
\end{align*}
```

$$\begin{aligned} B(r,\phi,\lambda) = \frac{\mu}{r}\Bigg[\sum_{n=2}^{\infty}\Bigg(&\left(\frac{R_e}{r}\right)^n J_n P_n(s\phi) \\ &+\sum_{m=1}^{n}\left(\frac{R_e}{r}\right)^n (C_{nm}\cos m\lambda + S_{nm}\sin m\lambda)P_{nm}(s\phi)\Bigg)\Bigg] \end{aligned}$$

별행 수식과 본문과의 간격을 비교해봅시다. 다음과 같은 길이 변수의 설정이 있다고 합시다.

```
\setlength\abovedisplayshortskip{0pt plus 2pt}
\setlength\belowdisplayshortskip{0pt plus 2pt}
\setlength\abovedisplayskip{20pt plus 5pt minus 3pt}
\setlength\belowdisplayskip{20pt plus 5pt minus 3pt}
```

이 문단처럼 본문이 별행 수식보다 길게 끝나면

$$f(x) = \int \frac{\sin x}{x}\,\mathrm{d}x \tag{1}$$

본문과 수식 사이의 수직 간격이 좀 벌어지게 설정되고, 본문이 다음 별행 수식보다 짧으면

$$f(x) = \int \frac{\sin x}{x}\,\mathrm{d}x \tag{2}$$

본문과 수식 사이의 간격이 좁아집니다. 눈으로 보이나요?

* * *

수식에 관한 훌륭한 지침서가 있습니다. `mathmode.pdf`를 읽어보세요.

```
> texdoc mathmode
```

## 2.2 표

LaTeX에서 표를 넣는 기본적인 방법은 `tabular` 환경을 쓰는 것입니다. 이때 최소한 자신이 작성하여야할 표의 열(column)이 몇 칸인지는 알아야합니다. 그리고 각 열에 들어갈 내용을 어떻게 정렬할 지는 알아야합니다. 다음 간단한 예제를 보시지요.

```
\begin{tabular}{l|c|r} \hline
\textsf{左} & \textsf{中} & \textsf{右} \\ \hline
왼 & 가 & 오 \\
왼쪽 & 가운 & 오른 \\
왼쪽 정 & 가운데 & 오른쪽 \\
왼쪽 정렬 & 가운데 정렬 & 오른쪽 정렬 \\ \hline
\end{tabular}
```

| 左 | 中 | 右 |
|---|---|---|
| 왼 | 가 | 오 |
| 왼쪽 | 가운 | 오른 |
| 왼쪽 정 | 가운데 | 오른쪽 |
| 왼쪽 정렬 | 가운데 정렬 | 오른쪽 정렬 |

수평선은 `\hline`으로, 수직선은 `|`로 긋습니다. 이렇게 단순하게 작성된 표에서 열의 폭은 열에 들어간 내용 중 가장 긴 내용에 맞추어 정해집니다. 이 예제에서는 1열은 '왼쪽 정렬' 2열과 3열은 '가운데 정렬'과 '오른쪽 정렬'의 내용이 가장 길군요. 매우 간단하지요? 그런데 문제는 이렇게 `c`, `l`, `r`로 열의 속성을 지정하면 해당 칸의 내용이 아무리 길더라도 자동으로 줄 바꿈이 되지 않는다는 것입니다. 다음 예제를 보시지요.

```
\begin{tabular}{l|c|r} \hline
\textsf{左} & \textsf{中} & \textsf{右} \\ \hline
왼 & 가 & 오 \\
왼쪽 & 가운 & 오른 \\
왼쪽 정 & 가운데 & 오른쪽 \\
왼쪽 정렬 & 가운데 정렬 & 오르으으으으으으으으으은쪽 정렬  \\ \hline
\end{tabular}
```

| 左 | 中 | 右 |
|---|---|---|
| 왼 | 가 | 오 |
| 왼쪽 | 가운 | 오른 |
| 왼쪽 정 | 가운데 | 오른쪽 |
| 왼쪽 정렬 | 가운데 정렬 | 오르으으으으으으으으으은쪽 정렬 |

뭡니까, 이게? `tabular` 환경 나빠요. 그래서 array 패키지를 쓰면 몇 가지 인자를 써서 열의 폭을 지정할 수 있습니다.

```
\usepackage{array}
...
\begin{tabular}{b{2cm}|m{3cm}|p{4cm}} \hline
\textsf{左} & \textsf{中} & \textsf{右} \\ \hline
사나이 가슴에 불을 당겨 & 사나이 가슴에 불을 당겨 & 오르으으으으으으으으은쪽 정렬  \\ \hline
\end{tabular}
```

| 左 | 中 | 右 |
|---|---|---|
| 사나이 가슴에 불을 당겨 | 사나이 가슴에 불을 당겨 | 오르으으으으으으으으으은쪽 정렬 |

오호라, 원하는대로 정해준 폭의 열이 생겨나고 그에 따라 해당 칸에서 내용이 줄바꿈이 되는군요. 그런데…뭔가 이상합니다. `b`, `m`, `p` 인자를 주어 칸의 폭을 정하고 줄 바꿈이 되는 것은 좋은데, 내용이 수직으로 위/ 가운데/ 아래에 정렬될 뿐이군요. 아까 처음에 했었던 왼쪽/ 가운데/ 오른쪽 정렬은 어떻게 된건가요? 예를 들어 내용 줄 바꿈도 하고 싶고 왼쪽 정렬을 하고 싶을 경우 말이에요.

```
\usepackage{array}
...
\begin{tabular}{>{\raggedright\arraybackslash}b{2cm}|
    >{\centering}m{3cm}|>{\raggedleft\arraybackslash}p{4cm}} \hline
\textsf{左} & \textsf{中} & \textsf{右} \\ \hline
사나이 가슴에 불을 당겨 & 사나이 가슴에 불을 당겨
    & 오르으으으으으으으으은쪽 정렬  \\ \hline
\end{tabular}
```

| 左 | 中 | 右 |
|---|---|---|
| 사나이<br>가슴에 불을<br>당겨 | 사나이 가슴에 불을<br>당겨 | 오르으으으으으으으으으<br>으은쪽 정렬 |

성공입니다! 처음에 왼쪽/ 가운데/ 오른쪽 정렬에 관한 명령어를 결합하여 열에 할당하면 되는군요. \raggedright, \centering, \raggedleft를 >과 결합하여 썼더니 잘 됩니다. 기분은 좋은데 어째 모르는 명령어가 있네요. \arraybackslash 명령은 뭡니까? 표에 직접적인 영향을 미치는 것은 아닌 것 같은데…. 사실은 LaTeX의 \raggedright, \raggedleft 명령어가 표의 행 나눔 인자인 \\을 재정의합니다. 그리하여 위의 표에서 \arraybackslash를 삽입하지 않으면 컴파일하는 데 애를 먹기 때문에 이를 바로잡기 위해서 삽입하였습니다. 이 명령어가 길다고요? 그럼 다음과 같이 해보세요. \\ 대신 \tabularnewline를 쓰는 것이지요.

```
\begin{tabular}{>{\raggedright}b{2cm}|>{\centering}m{3cm}|
    >{\raggedleft}p{4cm}} \hline
\textsf{左} & \textsf{中} & \textsf{右} \tabularnewline \hline
사나이 가슴에 불을 당겨 & 사나이 가슴에 불을 당겨
    & 오르으으으으으으으으으은쪽 정렬  \tabularnewline \hline
\end{tabular}
```

| 左 | 中 | 右 |
|---|---|---|
| 사나이<br>가슴에 불을<br>당겨 | 사나이 가슴에 불을<br>당겨 | 오르으으으으으으으으으<br>으은쪽 정렬 |

여기서는 행 나눔 명령인 \\ 대신 \tabularnewline을 썼습니다. 결과는 아까 표와 동일합니다. 이렇게 한 열 한 열의 폭을 지정하는 것은 알겠는데 전체 표의 폭을 정해줄 수는 없을까요? tabular* 환경을 쓰면 됩니다. 먼저 표의 전체 폭을 본문 가로 길이인 \textwidth로 정해볼까요? 이렇게 하면 상대적인 길이를 갖게 되어 부득이한 사정으로 문서의 크기를 바꾸어도 표의 길이에 신경 쓸 필요가 없겠네요.

```
\begin{tabular*}{\textwidth}{>{\raggedright}b{.3\textwidth}|
    >{\centering}m{.3\textwidth}|>{\raggedleft}p{.4\textwidth}} \hline
\textsf{左} & \textsf{中} & \textsf{右} \tabularnewline \hline
사나이 가슴에 불을 당겨 & 사나이 가슴에 불을 당겨
    & 오르으으으으으으으으으은쪽 정렬  \tabularnewline \hline
\end{tabular*}
```

| 左 | 中 | 右 |
|---|---|---|
| 사나이 가슴에 불을 당겨 | 사나이 가슴에 불을 당겨 | 오르으으으으으으으으으은쪽 정렬 |

야호 트랄랄라라! 성공입니다…. 그런데 가슴 한 구석이 또 허전하네요. 뭐가 문제일까요? 세 칸의 폭을 합친 길이가 분명 표 전체의 가로 폭인 본문 가로 길이와 같도록 했거든요. 근데 왜 삐져나가! (버럭)

$$
\begin{aligned}
&\text{첫 번째 열의 폭} + \text{두 번째 열의 폭} + \text{세 번째 열의 폭} \\
&= 0.3\backslash\texttt{textwidth} + 0.3\backslash\texttt{textwidth} + 0.4\backslash\texttt{textwidth} \\
&= 1.0\backslash\texttt{textwidth}
\end{aligned}
$$

이것은 표의 열과 열 사이에 기본적인 간격이 있기 때문입니다. 이 간격은 길이변수 \tabcolsep에 정의가 되어 있습니다. 이 길이를 없애주면 위 수식이 성립할 수도 있겠지요.

```
\setlength{\tabcolsep}{0pt}
\begin{tabular*}{\textwidth}{>{\raggedright}b{.3\textwidth}|
    >{\centering}m{.3\textwidth}|>{\raggedleft}p{.4\textwidth}} \hline
\textsf{左} & \textsf{中} & \textsf{右} \tabularnewline \hline
사나이 가슴에 불을 당겨 & 사나이 가슴에 불을 당겨
    & 오르으으으으으으으으으은쪽 정렬  \tabularnewline \hline
\end{tabular*}
```

| 左 | 中 | 右 |
|---|---|---|
| 사나이 가슴에 불을 당겨 | 사나이 가슴에 불을 당겨 | 오르으으으으으으으으으은쪽 정렬 |

* * *

이 정도만 알면 표를 사용하는 데 큰 문제는 없겠습니다. 그러나 더 복잡한 표를 만들어야할 경우도 생길 것입니다. 가령 표에 대각선을 넣거나 머리선과 바닥선의 두께를 굵게 하고 싶을 때, 위 행과 아래 행을 합치거나 한 쪽을 넘어가는 계속되는 표를 만들어야 할 때가 올 것입니다. KTUG FAQ 위키에 수 많은 표 관련 해법이 있습니다. Tabular 환경 페이지를 참조하여 tabuarx, tabuary, booktabs, multirow, slashbox, longtable, ltablex 같은 패키지를 쓰면 더 읽기 쉽고 보기 좋은 표를 만들 수 있습니다. 최근 개발된 tabu 패키지를 살펴보는 것도 권장합니다.

또한 memoir나 oblivoir 클래스에 포함된 수 많은 표에 관한 기능은 앞서 열거한 패키지를 불러오지 않아도 표를 작성하는 데 드는 수고를 덜어줄 것입니다.

## 3 더 읽어야 할 것

TEX으로 문서 작업을 꾸준히 하기 위해서 참고문헌을 읽어야 하는 것은 필수적입니다. 참고문헌만으로 해결할 수 없는 것은 KTUG[5] 게시판에 질문하시면 함께 토론해볼 수 있습니다. 여기서는 특히 한글 문서 작성과 관련하여 꼭 읽어야 할 문헌 몇 가지를 소개하겠습니다.

(1) LATEX 문서 작성에 대한 개괄적인 소개는 lshort-kr을 참고하십시오. `texdoc lshort-kr`.

(2) XELATEX을 이용한 한글 문서 작성에 대한 사항은 XETEXKO MANUAL을 꼭 읽어야 합니다. `texdoc xetexko`.

(3) oblivoir 클래스에 관해서는 세 곳을 참조하는 것이 좋습니다.

- 우선 memoir 매뉴얼을 읽어야 하는데 한국어 번역본이 있습니다. `texdoc memucs-manual`. 그러나 이것은 좀 이전판이므로 영문판을 읽으시기를 권합니다. `texdoc memoir`.
- http://faq.ktug.or.kr/faq/KTUG/Oblivoir/FAQ 페이지에 몇 가지 유용한 정보가 있습니다.
- XETEXKO, LUATEXKO manual을 반드시 읽어야 합니다.
- 〈초간단 설명서〉도 읽어두어야 합니다. `texdoc ultrasimplexob`.

(4) 수학식에 대해서는 Mathmode라는 문서를 권장합니다. `texdoc mathmode`. 당연히 AMS-LATEX 전반에 대한 소개로 AMS-LATEX document를 먼저 읽어두는 것이 좋습니다. `texdoc amsmath`.

항상 책이라는 것이 프로그램 자체의 발전보다는 더딜 수밖에 없어서 문서로 되어 있는 내용 중에는 최신의 코딩 방식이라든가 패키지 같은 것이 잘 소개되어 있지 않을 수 있습니다. 이에 대한 정보는 KTUG에서 토론하면 좋겠습니다.

[5] http://www.ktug.or.kr

## 찾아보기


유틸리티
- XeLaTeX, 2
- dvipdfm, 2, 5, 6
- dvipdfmx, 5
- dvips, 5
- ebb, 5, 6
- extractbb, 5, 6
- KC2008Plus, 2
- KCmenu, 2
- ko.TeX Live, 2
- latex, 3, 5
- LuaTeX, 2
- luatexko, 12
- Notepad++, 2
- pdfLaTeX, 2, 3
- pdfTeX, 6
- pstricks, 2
- xbb, 5, 6
- xeLaTeX, 3
- xetexko, 12
- xetexko manual, 12

패키지
- array, 9
- booktabs, 11
- epsfig, 3
- graphicx, 3, 5
- longtable, 11
- ltablex, 11
- memoir, 11, 12
- multirow, 11
- oblivoir, 11, 12
- psfig, 3
- pstricks, 5
- slashbox, 11
- tabu, 11
- tabuarx, 11
- tabuary, 11

형식
- .bb, 6
- .xbb, 6
- bb, 5, 6
- dvi, 2, 3
- eps, 3, 5, 6
- foo.tex, 3
- jpg, 3, 5
- mathmode.pdf, 8
- pdf, 2, 3, 5, 6
- png, 3, 5, 6
- ps, 2
- sample.bb, 5
- sample.jpg, 5
- sample.pdf, 3
- sample.xbb, 5
- tex, 3
- xbb, 5, 6