KC2008Plus 설치 후 첫 예제

KTUG Collection Team

October 16, 2010

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 [F6]을 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.

다른 문의사항은 http://www.ktug.or.kr/의 묻고답하기 게시판을 이용해 주십시오.

감사합니다. 즐텍 Happy TEX'ing.

KC2008Plus 설치 후 첫 예제

KTUG Collection Team

October 16, 2010

Abstract

KC2008이 설치되었습니다. KC2008은 TeXLive가 설치되어야 잘 동작합니다. 여러분은 지금 KC2008에 포함된 Notepad++을 통해 이 문서의 소스를 보고 계십니다.

- 이 문서의 `pdf` 파일을 얻으려면 F6 을 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.
- 이 문서의 `dvi` 파일을 얻으려면 Alt - F6 으로 LaTeX 컴파일 후 Dviout으로 열람하십시오. Dviout은 직접 명령행에서 실행해야 합니다. Dviout은 KC2008 설치 후 TeXLive를 업데이트하면 설치됩니다. `dvi`를 직접 화면으로 보는 것은 불가능하지 않지만 권장하지도 않습니다. Dviout에서는 이 문서의 삽입 그림이 보이지 않을 것입니다. Alt - F5 또는 Alt - F7 을 누르면 dvipdfm$x$를 거쳐 `pdf`를 볼 수 있습니다.
- XeLaTeX으로 컴파일하여[1] `pdf`를 볼 수도 있습니다. XeLaTeX컴파일을 위해서는 Ctrl - Alt - F6 또는 Ctrl - Alt - F5 를 누릅니다. OpenType 또는 TrueType 폰트를 사용합니다. 이 예제 문서에서 라틴 폰트는 로만 계열로 Palatino Linotype, 산세리프 계열로 Lucida Sans, 모노스페이스 계열로 Courier New를 사용하도록 했습니다. 한글 폰트는 은 글꼴 TrueType을 사용하는데 명조 계열로 은 바탕, 고딕 계열로 은 돋움, 타자 계열로 은 타자를 씁니다.

```
> xelatex first
```

- LuaTeX으로 컴파일하여[2] `pdf`를 볼 수도 있습니다.

[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.

```
> pdflualatex first
```

- 다른 문의사항은 http://www.ktug.or.kr/의 설치운영 게시판을 이용해 주십시오.

감사합니다. 즐텍 Happy TEX'ing.

1 돌다리 두드리기

최종 출력물 얻기

LaTeX 컴파일의 최종 출력물을 무엇으로 삼을 것이냐에 따라 몇 가지 방법이 있습니다. 전에는 `dvi`가 최종 출력물일 때도 있었지만 pdfTeX, ConTeXt의 등장으로 `pdf`가 최종 출력물로 간주되는 경우가 많아졌습니다. 이 글에서는 `pdf`를 최종 출력물의 포맷으로 보겠습니다. 우리가 컴파일하고자 하는 `tex` 소스의 파일 이름을 `foo.tex`이라 하지요.

dvi2pdf 간단하게 다음과 같이 하면 됩니다.

```
> latex foo
> dvipdfmx foo
```

pdf pdfLaTeX을 이용하여 직접 얻는 방법이 있습니다.

```
> pdflatex foo
```

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

```
> pdflatex -synctex=-1 foo
```

또는 dvipdfm$x$를 위하여 그림 크기를 미리 알도록

```
> latex -shell -synctex=-1 foo
```

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

그림 삽입

LaTeX에 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. 이렇게 상자로 인식하기 위해서 그림의 바깥 테두리의 크기(bounding box; `bb`)를 불러옵니다. `\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`이므로, pdfLaTeX이나 dvipdfm$x$가 이해하는 그림이면 쉽게 처리될 것입니다.

pdfLaTeX pdfLaTeX를 이용하여 `pdf`를 얻을 때는 삽입할 수 있는 그림이 많습니다.

- pdfLaTeX을 이용할 때는 pdf와 jpg, png를 삽입할 수 있습니다. 흑백 그림인 jbig2도 이용가능합니다. eps는 직접 삽입할 수 없지만 MetaPost eps인 mps는 잘 됩니다.

dvipdfm*x* dvipdfm*x*를 이용할 때는 LaTeX 실행 명령에 `-shell` 옵션을 주는 것만 잊지 않으면 됩니다. eps를 포함하여 jpg, png 등 여러 그림을 넣을 수 있습니다. 다만 pdf 그림을 불러오려면 xcolor가 필요한 경우가 있습니다.

```
> latex -shell foo
> dvipdfmx foo
```

그러면 그림을 하나 넣어볼까요? `sample.pdf`입니다. (6쪽의 Figure 1 참조) 그림 이름에 확장자를 쓰지 않은 것을 주의해서 보십시오.

```
\begin{figure}
\hangcaption
\captionnamefont{\small\sffamily}
\captiontitlefont{\small}
 \centering{%
\includegraphics[width=.75\textwidth]{sample}}
\caption{ Garamond 폰트의 f-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.

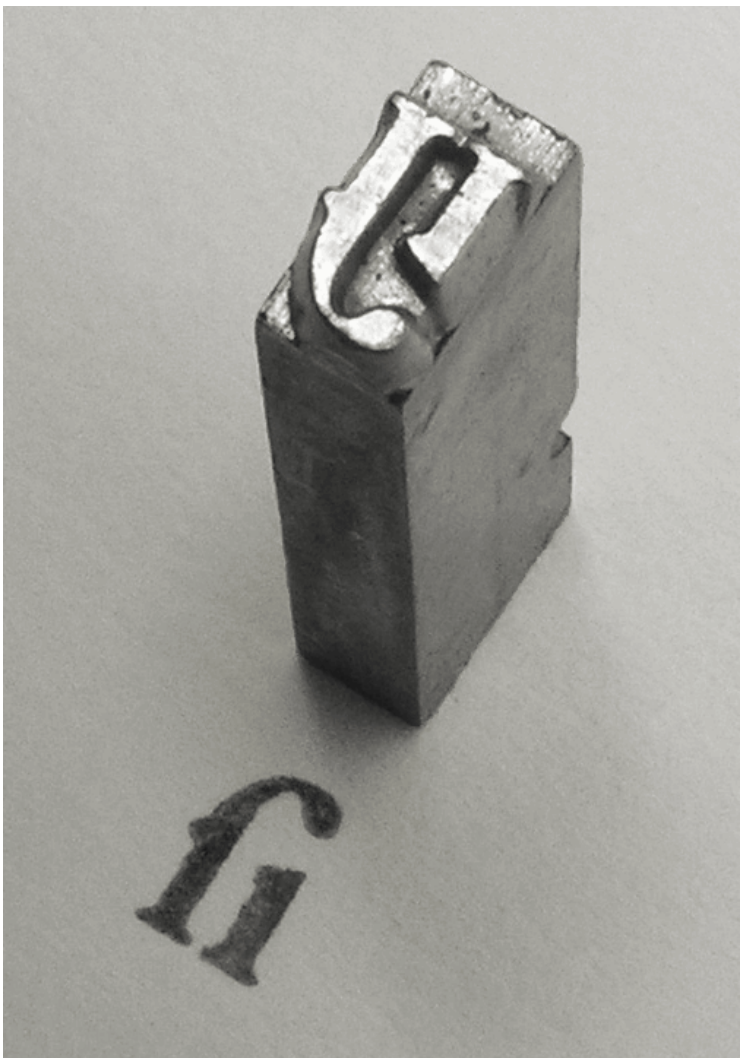

Figure 1: Garamond 폰트의 long s-i 리거처 (출처: WIKIPEDIA)

- 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.

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 DVIPDFMx 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 .xbb는 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 .bb입니다. .bb를 얻으려면 EXTRACTBB에 -m 옵션을 주고 실행합니다. EXTRACTBB의 기본값은 .xbb입니다. 과거에 만들어진 문서의 호환성을 위해서가 아니라면 특별히 EBB를 실행하여 .bb를 만들어 써야 할 필요는 없을 것입니다.

그러나… 모두 잊읍시다. eps 그림이 애를 먹이는 것도, 그림을 넣기 위해 앞에서 했던 bb 또는 xbb를 얻는 복잡한 과정도 모두 잊읍시다. 그림은 pdf나 png로 준비하면 됩니다. 또한, TEXLIVE 2008을 기반으로 한 KC2008은 graphicx 패키지의 옵션으로 dvipdfmx를 지정한 후 LATEX 명령 실행 시에 -shell 옵션을 추가하면 필요한 bb 또는 xbb를 직접 만들어 줍니다. 이렇게 하면 DVIPDFMx가 처리하는 jpg, png 등 그림의

크기가 pdfTEX으로 처리한 것과 동일한 결과를 얻습니다. 예전에 ebb를 이용할 때는 dvipdfm$x$ 가 pdfTEX에 비해서 그림을 더 크게 불러오도록 처리했었다고 합니다.

* * *

처음 사용자들이 자주 하는 질문 중에, 왜 LATEX과 dvipdfm$x$ 로는 되는데 pdfLATEX으로는 에러가 발생하느냐는 종류의 것이 있습니다. 이 문제에 대한 간단한 답은, pdfLATEX을 위해 준비된 그림과 소스는 pdfLATEX에서 안전하게 처리된다는 것입니다. 다른 컴파일 루트를 따르기로 한다면 그에 맞게 소스의 설정, 그리고 그림 준비 등을 알맞게 해야 하는 것입니다.

2 몇 가지 예제

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 $\left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right)$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 `\left`와 `\right`의 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.

```
> kctexdoc mathmode
```

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 & 오르으으으으으으으으으은쪽 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 줄바꿈이 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
    & 오르으으으으으으으으으은쪽 정렬  \\ \hline
\end{tabular}
```

左	中	右
사나이 가슴에 불을 당겨	사나이 가슴에 불을 당겨	오르으으으으으으으으으 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 가슴에 불을 당겨	사나이 가슴에 불을 당겨	오르으으으으으으으으으 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.

또한 memoir나 oblivoir 클래스에 포함된 수 많은 표에 관한 기능은 앞서 열거한 패키지를 불러오지 않아도 표를 작성하는 데 드는 수고를 덜어줄 것입니다. 더 자세한 것은 피터 윌슨(Peter Wilson) 씨의 memior 매뉴얼(제7판)이나 김강수 님이 번역한 우리말 memoir 매뉴얼(제6판)을 읽어보십시오.

```
> kctexdoc memman
> kctexdoc memucs-manual
```