The Not So Short Introduction to LaTeX 2ε

Chapter 6

LaTeX을 자기에게 맞게 바꾸기

Advanced Topic

KINS

2013년 1월 11일

제 6장

LaTeX을 자기에게 맞게 바꾸기

새로운 환경 정의하기

- LaTeX의 철학은 *"글쓴이는 문서의 논리적 구조에만 집중하라."* 이다.
- 따라서 문서의 본문에서 어떤 모양을 직접 바꾸는 명령을 쓰는 것은 좋은 생각이 아니다.
- 대신 문서의 모양을 바꿔야 할 일이 있으면 '새로운 명령'이나 '환경'을 정의해서 사용하는 것이 좋다.

새로운 환경 정의하기

새로운 환경은

```
\newenvironment{ name }[ n ][ i ]{ before }{ after }
```

형식으로 정의한다.

- name인자는 환경의 호출이름이다.
- n은 환경이 가질 수 있는 변수의 갯수이다. 각 변수는 환경을 정의할 때, #1, #2, ... 와 같이 이름붙여지며, 환경이 호출되어 이 변수에 대한 값이 주어지면 각각 #1, #2, ... 이 쓰인 곳을 그 값이 치환하게 된다.
- i는 초기값이다. 이 값이 선언되면, 첫 번째 변수는 선택항목이 되며, 환경을 호출할 때, 이 값을 생략하면 그 선택항목의 값을 i 로 취급한다.
- before과 after는 각각 환경이 호출되었을 때, 환경 내부 텍스트의 앞과 뒤에서 실행되는 명령들이다.

새로운 환경 정의하기 - 예시

```
\newenvironment{sol}[2][]
{\ignorespaces \textbf{문
제 #1.#2. }}
{\hspace{\stretch{1}}$\square$
 \ignorespacesafterend}

\begin{sol}{2}{3}
Easy. Do it yourself.
\end{sol}

\begin{sol}{2}{4}
Trivial.
\end{sol}
```

문제 2.3. Easy. Do it yourself. □

문제 2.4. Trivial. □

Remark

- 새로 정의할 환경의 이름이 이미 정의되어 있는 경우 에러가 발생한다.
- 이 때, 사용자 정의 환경의 이름을 바꾸던가 아니면

  `\renewenvironment{ name }[ n ]{ before }{ after }`

  명령을 써서 재정의해주면 된다. 이 때, 기존의 정의는 무시된다.

새로운 명령 정의하기

새로운 명령을 정의하는 방법 역시 거의 같다.

```
\newcommand{ name }[ n ] [ i ]{ definition }
\renewcommand{ name }[ n ][ i ]{ definition }
```

- name은 명령의 이름이다. 백슬래시\을 꼭 포함해야 한다.
- n은 명령이 가지는 인자의 갯수이다.
- i가 선언되면 첫 번째 변수는 선택사항이 되며 명령 호출 시 값이 주어지지 않으면 i로 치환된다.
- definition은 명령이 호출되었을 때 할 일이다.

새로운 명령 정의하기

```
\renewcommand{\C}[2][\infty]
{C^{#1}(\mathbb{#2})}

Let $\C{R}$  be a set of
smooth functions
and $\C[1]{R}$ be a
set of continuously
differentible functions  on
$\mathbb{R}$.
```

Let $C^{\infty}(\mathbb{R})$ be a set of smooth functions and $C^{1}(\mathbb{R})$ be a set of continuously differentible functions on $\mathbb{R}$.

사용자 정의 스타일

새로운 명령어나 환경, 패키지를 너무 많이 사용한다면 이들을 한데 묶어 새로운 패키지로 만드는 것이 좋다.

이는 필요한 패키지와 새로 정의한 환경, 명령어를 그대로 텍스트 파일에 복사한 후 확장자를 .sty로 바꾸면 된다.

이렇게 만든 새로운 패키지는 `\usepackage{...}`명령을 써서 불러올 수 있다.

길이 문제

임의의 수평 간격과 수직 간격을 줄 수 있는 명령어가 있다.

- `\hspace{ length }`는 길이가 length인 수평 공백을 삽입해주고,
- `\vspace{ length }`는 길이가 length인 수직 간격을 *행 사이에* 삽입해준다.
- `\hspace{\stretch{1}}`명령은 명령이 쓰인 줄의 끝까지 수평 공백을 채워넣어준다.

길이 문제

이 때, length에는 *길이변수의 배수*, 또는 *길이+단위*가 와야 한다.

LaTeX에서는 다음과 같은 단위가 쓰인다.

mm	1mm=
cm	1cm=
pt	1pt=
in	1in =
em	1em=현재 폰트에서 글자 M의 너비=
ex	1ex=현재 폰트에서 글자 x의 너비=

길이 문제

또한, 대표적인 LaTeX에서 미리 정의된 길이 변수는 다음과 같다.

- `\textwidth`: 현재 서식의 표준 문단 폭
- `\parindent`: 현재 서식의 문단 들여쓰기 간격
- `\parskip`: 현재 서식의 문단 사이의 간격
- `\baselineskip`: 현재 서식의 행간격

어떤 길이 관련 문제를 접했을 때, 웬만하면 ***절대길이*(mm, cm, pt, in)*는 사용하지 않는 게 좋다.*** 문서 서식이 바뀌면 이들 절대 길이가 사고를 치는 일이 많기 때문이다. 대신 가변길이 단위인 em, ex 또는 길이 변수들을 쓰는 것이 좋겠다.

길이 변수

LaTeX에서 미리 정의된 길이 변수 외에 사용자가 새로 길이 변수를 다음 명령어를 사용해서 정의할 수도 있다.

```
\newlength{ name }
```

이 때, name은 길이 변수의 이름으로 백슬래시 \를 포함해야 한다.

길이 변수

LaTeX의 미리 정의된 길이 변수나 사용자가 만든 길이 길이 변수는 다음과 같은 명령어로 조절할 수 있다. 아래 표에서 var은 길이 변수를 말하고, val는 숫자+단위를 말한다.

명령어	설명
`\setlength{var}{val}`	var에 val를 값으로 할당한다.
`\addtolength{var}{val}`	var의 현재 값에 val을 더한다.
`\settoheight{var}{text}`	문자열 text의 높이를 var의 값에 할당한다.
`\settowidth{var}{text}`	문자열 text의 폭을 var의 값에 할당한다.
`\settodepth{var}{text}`	문자열 text의 깊이를 var의 값에 할당한다.

Remark

LaTeX의 모든 레이아웃은 미리 정의된 길이 변수들을 가지고 구성된다. 따라서 이미 정의된 길이를 바꾸면 문서의 모양 역시 바뀐다.

예를 들어, `\setlength`명령을 써서 `\textwidth`의 값을 바꾸면 문서의 문단 폭이 바뀌는 효과를 가져온다. 따라서, 이들 값은 필요한 경우가 아니면 함부로 바꾸지 말자.

길이 변수

```
\setlength{\parindent}{1em}
\setlength{\parskip}{2em}
이제는 문단 들여쓰기가 됩니
다. 이 문단은 들여쓰기가 되었
죠.

한편 문단 사이의 간격도 이렇
게 벌어졌네요.
```

이제는 문단 들여쓰기가 됩니다. 이 문단은 들여쓰기가 되었죠.

한편 문단 사이의 간격도 이렇게 벌어졌네요.

- 연습문제 1. 다음을 조판하시오.

안녕하세요. 이 문단은 첨삭을 위해 double space를 적용하여 쓰여졌습니다. □

페이지 레이아웃

페이지 레이아웃을 바꾸는 방법을 알아보자.

- LaTeX의 페이지 구성은 lshort 119페이지에 나와 있다.
  http://faq.ktug.or.kr/wiki/uploads/layout.pdf
- LaTeX은 짝수 페이지와 홀수 페이지의 구성이 조금 다르다.
  twoside옵션을 지원하지 않는 문서클래스(article)의 경우 기본적으로 홀수 페이지의 레이아웃을 따른다.
- 모든 레이아웃의 파라미터는 길이 변경 명령어 `\setlength`, `\addtolength`등을 사용해 바꿀 수 있다.

Remark

LaTeX의 페이지 레이아웃은 가독성을 고려하여 신중하게 정해진 것이다. 너무 좁다는 이유만으로 레이아웃을 건들이면 문서의 가독성이 떨어질 수 있으니 신중하자.

박스

- 박스는 LaTeX이 문서를 만들어가는 기본 단위이다.
- 박스 안에는 다른 박스를 포함한 어떤 것이던 들어갈 수 있다.
- 같은 박스에 있는 객체는 마치 '한 몸'인 것처럼 행동한다.
- ***하나의 박스는 하나의 '글자'로 취급***되며 실제 출력물에서 박스가 차지하는 크기는 (박스 안에 들어 있는 내용물과 무관하게) 박스를 정의할 때 정해진다.

직접 언급하진 않았지만 tabular, includegraphics, picture등의 명령/환경은 모두 박스를 생성한다.

박스

사용자가 박스를 직접 만들 수 있다. 다음 minipage

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

환경은 문단 전체를 하나의 박스로 만들어준다.

- pos인자는 baseline을 기준으로 박스의 위치를 결정해준다. t,c,b를 인자로 쓸 수 있다.
- width는 박스의 크기를 결정해준다. 이 크기는 박스 안의 내용물의 크기와는 무관하지만 표준 문단폭보단 작아야한다.
- 사실 환경 이름이 말해주듯, minipage환경은 작은 페이지를 만든다. 따라서 ***이 환경 내에서는 표준 문단폭이 width가 된다.***

박스 - 예시

```
\begin{minipage}{0.3\textwidth}
우리는 한 몸. 같은 박스에 들어 있지.
\end{minipage}
널 갈라놓겠어!
\begin{minipage}{0.3\textwidth}
우리도 한 몸. 우릴 갈라놓을 순 없지.
\end{minipage}
```

우리는 한 몸. 같은 박스에 들어 있지. 널 갈라놓겠어! 우리도 한 몸. 우릴 갈라놓을 순 없지.

박스

수평한 박스를 다른 박스에 넣는 방법도 있다.

```
\makebox[ width ][ pos ]{ text }
\framebox[ width ][ pos ]{ text }
```

위 명령은 수평으로 배열된 박스를 하나의 박스로 묶어준다. framebox는 박스의 테두리를 그려주는것만 빼면 makebox와 정확히 같은 기능을 한다.

- width는 박스의 폭을 결정한다. 역시 박스 내부에 들어 있는 텍스트의 크기와 무관하다.
- pos는 박스 안의 텍스트 배열을 결정한다. c(가운데 몰기), l(왼쪽 몰기), r(오른쪽 몰기), s(크기에 맞게 펼치기)를 사용할 수 있다.
- text는 박스에 넣을 텍스트이다.

박스 - 예시

```
\makebox[10em][c]{가운데 몰기}

\makebox[10em][s]{펼 치 기}

\framebox[5em][l]{박스는 넘칠 수 있다.} 내 영역을 침범하지
마.
```

가운데 몰기

펼 치 기

박스는 넘칠 수 있다. 내 영역을 침범하지마.

박스

박스의 수직 위치를 조절하는 명령어가 있다.

```
\raisebox{ lift }{ text }
```

를 사용하면 박스를 기준 위치에서 lift만큼 수직 이동시킬 수 있다.

```
Oh, \raisebox{2em}{My}
 \raisebox{-1em}{God!}
LaTeX
```

Oh, My God!

LaTeX

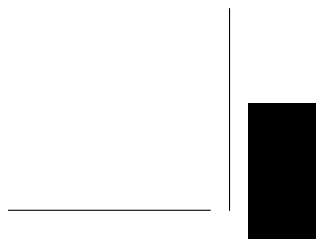
선 그리기

```
\rule[ lift ]{ width }{ height }
```

- 폭이 width이고 높이가 height인 검정 상자를 그려준다.
- lift는 기준 위치로부터 상자의 수직 위치를 결정한다.
- 폭과 높이를 잘 조절하면 이 명령어를 써서 수직선, 수평선을 그릴 수 있다.
- 폭을 0으로 주면 높이만 있는 투명한 선이 되는데, 셀의 높이를 최소 일정한 값 이상으로 유지하는데 사용된다.

선 그리기 - 예시

```
\rule{0.3\textwidth}{.2pt}
\rule{.2pt}{0.3\textwidth}
\rule[-5pt]{0.1\textwidth}
{0.2\textwidth}
```



- 연습문제 2. 다음 표를 그리시오.

1.	이 행은	높이가 높다.
2.	이 행은	높이가 낮다.

Advanced Topic

카운터

LaTeX이 '번호 붙는 항목'을 내부적으로 어떻게 처리하는지 알아보자.

카운터

모든 번호붙이기 항목은 LaTeX내부에서 **counter**라는 변수를 사용해 처리된다.

표준적으로 미리 정의된 counter이름

- chapter: 쳅터 번호를 저장하는 변수
- section: 섹션 번호를 저장하는 변수. chapter가 바뀔때마가 reset.
- subsection: 서브 섹션 번호를 저장하는 변수. section이 바뀔때마다 reset.
- figure, table: 그림, 표의 번호를 저장하는 변수
- enumi, enumii, enumiii, enumiv: `enumerate`환경에서의 항목 번호.
- footnote: 각주의 번호를 저장하는 변수.
- equation: 식 번호를 저장하는 변수.
- page: 페이지 번호를 저장하는 변수.

카운터

저장된 카운터를 출력하려면 다음과 같은 명령을 쓴다.

```
\thecounter
```

여기서 counter는 counter의 이름이 오는 자리이다. 예를 들어, `\thechapter`는 chapter에 해당하는 카운터를 출력해준다. `\thecounter`명령은 기본적으로 아라비아 숫자로 출력을 해 준다. 출력 모양을 지정하려면 다음과 같은 명령을 쓸 수 있다.

- `\arabic{counter}`: 현재 카운터에 저장된 번호를 아라비아 숫자로 출력.(1,2,...)
- `\Roman{counter}`, `\roman{counter}`: 현재 카운터에 저장된 번호를 로마 대문자 (I, II, ...) 또는 소문자(i,ii,..)로 출력.
- `\Alph{counter}`, `\alph{counter}`: 현재 카운터에 저장된 번호를 알파벳 대문자(A,B,..)또는 소문자(a,b,...)로 출력.

카운터

- \thecounter역시 명령어이므로 \renewcommand를 써서 얼마든 재정의할 수 있다.

```
\renewcommand{\theequation}{\thesection.\thesubsection.\arabic{equation}}
\begin{equation}
F=ma
\end{equation}
```

$$F = ma \tag{2.1.1}$$

- 연습문제 3. \theenumi명령어를 재정의해서 다음과 같은 항목나열을 식자하라.

  I. 항목 번호가

  II. 바뀜!

     II.i 여기도

     II.ii 바뀜!

  III. 계속 바뀜!

사용자 정의 카운터

기존에 있는 카운터 외에 사용자가 직접 새로운 카운터를 다음 명령으로 만들 수 있다.

```
\newcounter{ name }[ reset ]
```

- name은 카운터의 이름이다. 물론 기존의 것과 중복되면 안 된다.
- reset인자는 이미 정의된 다른 카운터의 이름이며, reset이라는 카운터가 바뀔때마다 새 카운터의 값을 0으로 초기화한다. 즉, name이라는 카운터를 reset이라는 카운터에 종속시킨다.

새로 만든 카운터는 `\setcounter{ name }{ value }`명령으로 값을 줄 수 있다. 이 작업을 하지 않으면 초기값 0으로 설정된다.

사용자 정의 카운터

카운터에 관련된 명령이 여럿 있다.

- `\stepcounter{ counter }`: counter의 값을 1 증가시킨다.
- `\refstepcounter{ counter }`: 현재 counter값을 1증가시킨 후 그 값을 `\label`에 저장한다.
- `\addtocounter{ counter }{ value }`: 현재 counter값에 value를 더한다.
- `\value{ counter }`: 현재 counter에 저장된 값을 '값으로서' 내보낸다. 이는 LaTeX의 다른 명령의 argument 내부에서 쓰인다.

사용자 정의 카운터 -예시

```
\newcounter{thm}[section]
\newenvironment{thm}{\ignorespaces \refstepcounter{thm}
\textbf{Theorem \thesection.\thethm.}}{\ignorespacesafterend}
\begin{thm}\label{this}
1+1=2.
\end{thm}

\begin{thm}
1+2=3.
\end{thm}

우리는 정리 \ref{this}에서, 다음을 알 수 있다.
```

Theorem 2.1. 1+1=2.

Theorem 2.2. 1+2=3.

우리는 정리 **??**에서, 다음을 알 수 있다.

- 숙제 3. 다음 문서를 조판하라.

프로젝트에 관하여

이번 세미나의 프로젝트는 지금까지 배운 테크닉을 활용하여 (손으로 쓰여진) 실제 강의노트의 일부를 LaTeX문서로 바꾸는 것입니다. 다음 사항을 고려해서 프로젝트를 진행하시면 됩니다.

- article클래스 pagestyle은 heading으로 해 주세요. 샘플로 주어진 강의노트는 하나의 section입니다.
- 강의노트를 보시며 최대한 가독성과 논리적 구조가 잘 살아나도록 만들면 됩니다. 지금까지 배운 모든 테크닉을 200% 활용하세요.
- 샘플 강의노트의 모양에 크게 구애받지 마세요. 논리적인 구조를 최대한 잘 살릴 수 있도록 얼마든지 모양을 바꾸셔도 됩니다.
- 실제 책은 손으로 쓴 강의노트와는 그 형식이 분명히 달라야 합니다. 예를 들어 노트에는 example의 numbering이 없지만 책으로 만들 경우 numbering이 분명 필요할 것입니다.