모두를 위한 LaTeX

부제 : 근본없는 매뉴얼

dokenzy@gmail.com

0.6판
2014년 3월 27일

요 약

이 문서의 목적은 대략적인 LaTeX문서작성방법을 소개하고, 좋은 LaTeX 설명서를 소개하는 것이다. 이 문서를 읽고 LaTeX을 제대로 배우고 싶은 분들은 이 문서 마지막에 있는 8장 추천 문서를 읽기를 바란다.

[1] http://creativecommons.org/licenses/by/4.0/

바뀐 점

0.6 2014년 3월 25일. 특수문자 설명 추가, 참고문헌 추가, 자잘한 부분 수정

0.5 2014년 2월 18일. TeX Live 업데이트 내용 추가, 자잘한 부분 수정

0.4 2014년 2월 4일. 찾아보기 추가, defaultfontfeatures 내용 추가, 카운터 이름 변경, 추천문서 추가, 자잘한 것 몇 개 수정

0.3 2013년 12월 20일. 맞춤법 수정, MWE 내용 추가, 자잘한 것 몇 개 수정. 조언해주신 이주호님께 감사드립니다.

0.2 2013년 12월 20일. '워드 프로세서, 멍청하고 비효율적인 도구' 한글판 링크 추가, 자잘한 것 몇 개 수정

0.1 2013년 12월 13일. 최초 공개

차 례

들어가며

왜 LaTeX인가?

이 문서는 나 자신이 LaTeX[2]을 하며 느꼈던 점들을 바탕으로 만든 LaTeX초보용 문서이다. 나는 LaTeX뿐만 아니라 조판 및 인쇄 전반에 대한 실무경험이 전혀 없을 뿐 더러, 이론적 지식도 영화 '그라비티'에 나오는 사람 수만큼 밖에 안된다. 맞춤법도 그다지 자신 없다. 다만, 마크업과 일반 텍스트(plain text)를 동경하는 사람으로서 최소한의 노력으로 최대한 멋진 문서를 만들고 싶은 욕심이 있다.

나처럼 마크업과 일반 텍스트를 이용해서 문서를 만드는 사람이 많을 것이다. Markdown이나 reStructuredText 같은 것들이 개발자들에게 많이 쓰이는 이유이기도 하다. Markdown이나 reST가 생산적으로 문서를 만들어 주긴 하지만 멋진 문서를 만들어 주지는 않는다. 이를 충족시킬 수 있는 도구가 LaTeX이다. 매크로 언어인 LaTeX과 조판프로그램인 TeX을 이용하면 생산적이면서도 품위 있는 문서를 만들 수 있다.

워드프로세서와 무엇이 어떻게 다른지 궁금한 사람들을 위해 The Beauty of LaTeX이라는 글을 소개한다. 해석하지 않아도 된다. 단지 눈으로 비교해 보는 것만으로도 충분하다. 게다가 워드프로세서는 물론, 인디자인 같은 전문 조판 프로그램조차 상상도 할 수 없는 일을 할 수 있다. 2011년 한국텍학회 학술대회에서 조진환 교수님께서 발표하신 함초롬 - 글꼴을 이용한 자소 분리를 보라

[2]레이텍 또는 라텍이라 읽는다.

(그림 1). 이 발표는 2013년 10월에 열린 TUG 2013에서도 발표되어 큰 관심을 받았다.

CSS 없이 HTML 코드에 스타일을 구겨 넣었던 1990년대를 기억하는가? 그런데 워드프로세서는 왜 아직도 그 꼴을 벗어나지 못하는 것인가? 워드프로세서는 할 수 없다. 앞으로도 그렇다. 김강수님이 번역하신 워드 프로세서, 멍청하고 비효율적인 도구라는 글을 읽어보자. 나 또한 완전히 동의한다. 이제 워드프로세서의 늪에서 벗어나 보자.

그림 1 함초롬을 이용한 자소 분리(조진환)

한국텍학회와 한글텍사용자그룹

내가 이 문서를 만들 수 있게 된 것은 한국텍학회(KTS)[3]와 한글TEX사용자그룹(KTUG)[4]에서 활동하시는 몇몇 분들의 헌신적인 노력 덕분이다. 이분들은 ko.TEX[5]을 개발하셨을 뿐만 아니라 관련 문서들을 직접 쓰거나 번역하시고 정기적인 워크숍 등을 통해 무료로 교육까지 하신다. 그뿐만 아니라 TnX텍 같은 유용한 유틸리티 프로그램도 개발하셨다. 내가 LATEX을 배울 수 있었던 것은 모두 이분들의 공헌 덕분이다. 이분들은 TEX뿐만 아니라 타이포그래피, 레이아웃, 배치(batch)처리, 폰트 등 조판과 인쇄분야 전반에 해박한 지식과 풍부한 경험이 있다. 위에서 보인 '함초롬을 이용한 자소 분리(조진환)'(그림 1)에 쓰인 폰트는 '한글과컴퓨터'사에서 개발한 함초롬폰트에 한국텍학회가 GSUB/GPOS 정보를 추가하여 재배포하는 폰트다.[6]

2013년 11월 9일, 한국어 조판을 위한 ko.TEX이 TeXLive 2013에 드디어 정식으로 포함되었다. 이것은 당신이 만든 Python 라이브러리가 Python 표준 라이브러리에 등록된 것과 비슷하다. 물론 TeXLive에 포함되기 전에도 ko.TEX으로 한국어 문서를 조판할 수 있었지만, 이젠 TeXLive에 포함됨으로써 별도의 설치과정을 거치지 않아도 된다.

XHLATEX은 무엇인가?

이 절에서 언급하는 단어들을 처음 접하는 사람들은 혼란스러울 수 있다. 아니면 이미 등장한 LATEX, ko.TEX, TEX이란 용어때문에 아까부터 혼란스러운 사람이 있을 수도 있다. 조금이라도 혼란스럽다면 과감히 이 절을 넘어가자. 시작하기도 전에 용어 때문에 스트레스받지 말자. 자동차 운전을 처음 배우는 사람이 자동차 엔진부터 배울 필요는 없다. 이 문서는 운전을 시연하는 수준일 뿐이다. LATEX이라는 차를 어떻게 운전하는지 지켜본다는 마음으로 가볍게 읽자.

[3] The Korean TeX Society

[4] http://ktug.org

[5] '케이오텍' 또는 '코리언 텍'이라고 읽는다.

[6] 이 폰트에 대한 자세한 정보는 함초롬체LVT를 참고하라.

LaTeX은 매크로언어이다. LaTeX을 한다는 말은 이 매크로언어로 만든 텍 파일(*.tex)을 처리해서 최종적으로 원하는 문서(예를 들면, PDF파일)를 만든다는 것이다. 이 텍 파일을 처리하기 위해서는 텍 엔진이라는 프로그램이 필요하다. 현재 TeXLive에는 5개의 텍 엔진이 있는데 이 문서에서는 XeTeX이라는 엔진을 이용한 방법만 다룬다. XeTeX은 시스템에 설치된 대부분의 truetype이나 opentype 폰트 등을 직접 사용할 수 있기 때문이다.[7]

ko.TeX은 TeX으로 한글을 조판하기 위한 시스템이다. XeTeX-k*o*는 XeTeX 엔진을 위한 ko.TeX패키지이다. ko.TeX2.0 설명서에 따르면 현재 한글 문헌을 가장 안정적으로 조판할 수 있는 것은 XeTeX-k*o*라고 한다. 그래서 이 문서에서는 XeLaTeX만을 다룬다.[8] XeLaTeX이란 'XeTeX이라는 텍 엔진'으로 'LaTeX 매크로언어로 작성한 텍 파일'을 처리하는 프로그램이라고 할 수 있겠다.

[7] XeTeX에서 사용하지 못하는 트루타입이나 오픈타입 폰트는 대부분 폰트 자체에 문제가 있다.

[8] 내가 그나마 조금 할 줄 아는 것도 이것 뿐이다.

제 1 장

첫걸음

1.1 운영체제별 설치

LaTeX을 사용하기 위해서는 TeXLive 2013을 설치해야 한다.[1] TeXLive는 사실상의 표준 텍 시스템으로서 Mac OS X, 윈도우즈, 리눅스 운영체제에서 사용할 수 있다. 텍 시스템은 많은 유틸리티와 방대한 라이브러리, 스타일, 폰트 등으로 구성된 큰 시스템이다. MacTeX의 경우 설치파일만 2.3GB에 달한다. 전문가는 원하는 부분만 설치해서 사용하는 것도 가능하지만, 이 문서에서는 텍 시스템 전체를 설치하는 것만 다룬다.

Mac MacTeX 2013[2]을 설치한다. 이 배포판은 Mac 10.5(Leopard) 이상에서 사용할 수 있다. 인스톨러의 안내대로 설치하면 끝난다.

Windows 윈도우즈 사용자는 KTS와 KTUG에서 제작한 ko.TeXLive 2013[3]을 이용하면 쉽게 설치할 수 있다. 인스톨러의 안내대로 설치하면 끝난다.

[1] http://www.tug.org/texlive/

[2] http://tug.org/mactex/

[3] ko.TeX Live 2013

설치 중 일부 파일에 대해 백신 프로그램이 바이러스로 오진하여 설치가 중단될 수 있다. 이럴 때는 잠시 백신을 중지하고 설치하자.

Linux 리눅스에서는 인터넷을 통해 설치하는 방법을 추천한다. 설치페이지에서 install-tl-unx.tar.gz파일을 다운로드한 후 install-tl 설치 스크립트를 이용해서 설치한다. perl-tk가 설치되어 있다면 설치 스크립트를 실행할 때 --gui 옵션을 추가하여 GUI로 설치를 진행할 수 있다. 일부 배포판의 경우 개인 사용자들이 패키징을 해놓은 것들이 있다. 이런 패키지를 이용하는 방법도 있다.

TeXLive 업데이트 MacTeX이나 ko.TeXLive 2013으로 설치한 후에는 업데이트를 해주어야 한다. 또한 TeXLive를 사용하다가도 한 달에 한 번 정도는 업데이트를 해주는 것이 좋다. TeX Live를 최신 버전으로 업데이트하기 위해서는 명령행에서 다음 명령을 실행한다.

Code 1

```
$ tlmgr update --self --all
```

위 명령을 실행했는데 퍼미션 오류가 발생한다면 root권한으로 위 명령을 실행해야 한다. Mac OS X나 데비안 계열의 경우 sudo와 함께 사용하면 된다. 다른 리눅스에서도 root권한으로 실행하면 된다.

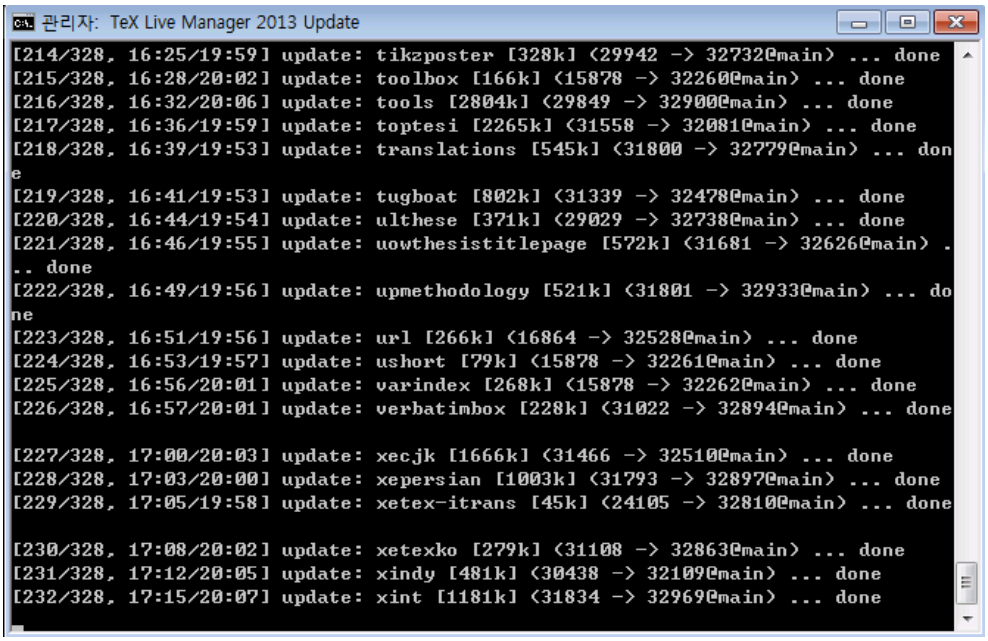
업데이트를 오랫동안 안하면 그림 1.1처럼 수백 개를 한 번에 업데이트하게 되는 수가 있다.

1.2 Mac OS X에서 폰트경로 설정하기

Mac OS에서는 XeTeX을 위한 설정을 한 번 해주어야 한다. [4]

[4] 윈도우즈는 `fc-cache`명령을 실행해야 한다. ko.TeX Live 2013는 설치 과정 마지막에 자동으로 해준다. 하지만 설치 후에 추가된 폰트를 사용하기 위해서는 이 명령을 직접 실행해 주어야 한다.

그림 1.1 tlmgr update



/usr/local/texlive/2013/texmf.cnf 파일을 텍스트 에디터로 열어서 다음 내용을 추가하자.

Code 2

```
OSFONTDIR = {~/Library/Fonts;/Library/Fonts;/System/Library/Fonts}
```

이 설정을 하면 시스템 라이브러리나 사용자 라이브러리 폴더에 있는 폰트들을 XeTeX이 사용할 수 있다. 자세한 내용은 KTUG wiki를 참고하자.

그리고 블로그 '도은이네 집'에서 소개한 맥 오에스 폰트 라이브러리 등록 글에 소개된 내용도 적용하자. TeXLive를 설치하면 많은 폰트들이 함께 설치되는데, 이 경로는 위에서 설정한 경로와 다르다. XeTeX이 이 폰트들을 사용하기 위한 방법이다.

1. 서체 관리자(Font Book)을 실행한다.

2. 파일-새로운 보관함(New Library) 메뉴를 선택하고 'TL OpenTypes'라고 이름을 지어주자.

3. 방금 만든 'TL OpenTypes'를 오른쪽 클릭하여 '서체 추가...'를 선택하자.

4. 파일 선택창이 열리면 '⇧+⌘+G'키를 눌러 경로창을 띄운 후 다음 주소를 입력하자.

   Code 3

   ```
   /usr/local/texlive/2013/texmf-dist/fonts/opentype/public
   ```

   굉장히 많은 폰트들이 각 폴더에 들어있다. 전부 다 해도 되고 원하는 폰트(폴더)만 선택해도 된다.

같은 방식으로 truetype 폰트들도 추가할 수 있다.[5] 이제 Mac에서 XeTeX을 사용하기 위한 설정을 모두 마쳤다[6].

1.3 테스트

이제 LaTeX을 사용하기 위한 준비를 모두 마쳤다. 아무 텍스트 편집기라도 사용할 수 있지만 여기서는 TeXLive를 설치할 때 함께 설치되는 TeXworks를 사용하기로 한다. TeXworks를 실행하고 아래 코드를 입력하자.

[5] /usr/local/texlive/2013/texmf-dist/fonts/truetype/public

[6] 윈도우즈에서는 폰트를 시스템에 설치할 때마다 fc-cache 명령을 실행하면 된다.

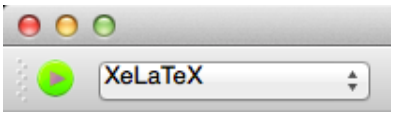
Code 4

```
\documentclass[oneside, a4paper]{article}
\usepackage{kotex}
\setmainhangulfont{HCR Batang LVT}
% 함초롬 바탕 LVT를 다운로드 & 설치하자
% 다운로드 주소: http://bit.ly/1hnXefL
\begin{document}
\section[시작]{\LaTeX 시작하기}
\begin{center}
안녕, \LaTeX!
\end{center}
한 번 배워서 오래 써먹자.
\end{document}
```

입력을 마치면 foo.tex이라는 이름으로 저장하자.

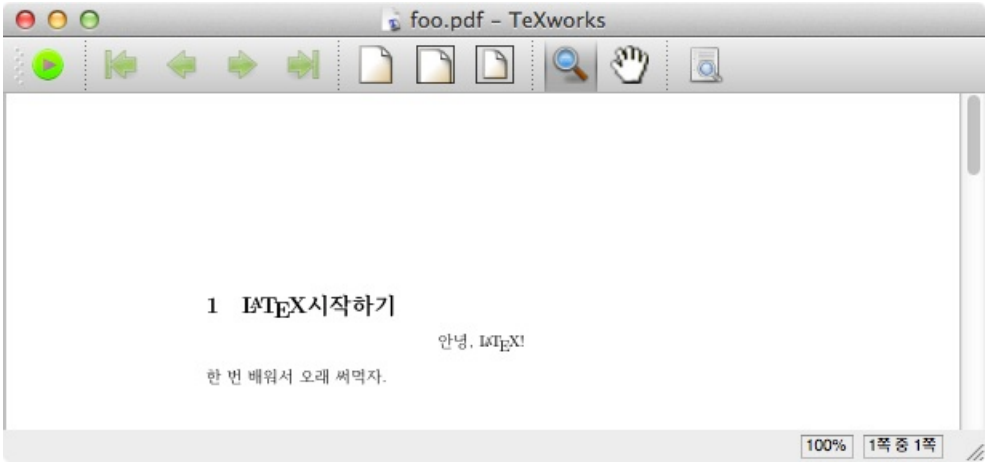
이제 텍 엔진을 그림 1.2와 같이 선택하고 컴파일 버튼을 누르거나 [조판] 메뉴에서 '조판'을 누르자. 또는 단축키를 이용할 수도 있다.

그림 1.2 TeXworks의 컴파일 버튼과 텍 엔진



1) TeXLive 2013이 제대로 설치되고, 2) 소스코드에 오류가 없고, 3) 텍 엔진을 올바르게 선택했다면 컴파일이 끝난후, 화면 오른쪽에 그림 1.3처럼 foo.pdf 파일이 보일 것이다.

그림 1.3 foo.pdf



나는 Skim[7]이라는 PDF리더를 주로 사용하지만, PDF파일의 정보를 볼 때는 Adobe사의 Acrobat Reader를 사용하기도 한다. Acrobat Reader로 foo.pdf 파일의 속성을 보면 그림 1.4처럼 나온다. XeTeX으로 만들었다는 것을 알 수 있다.

1.4 알고보면 사연이 많은 예제코드

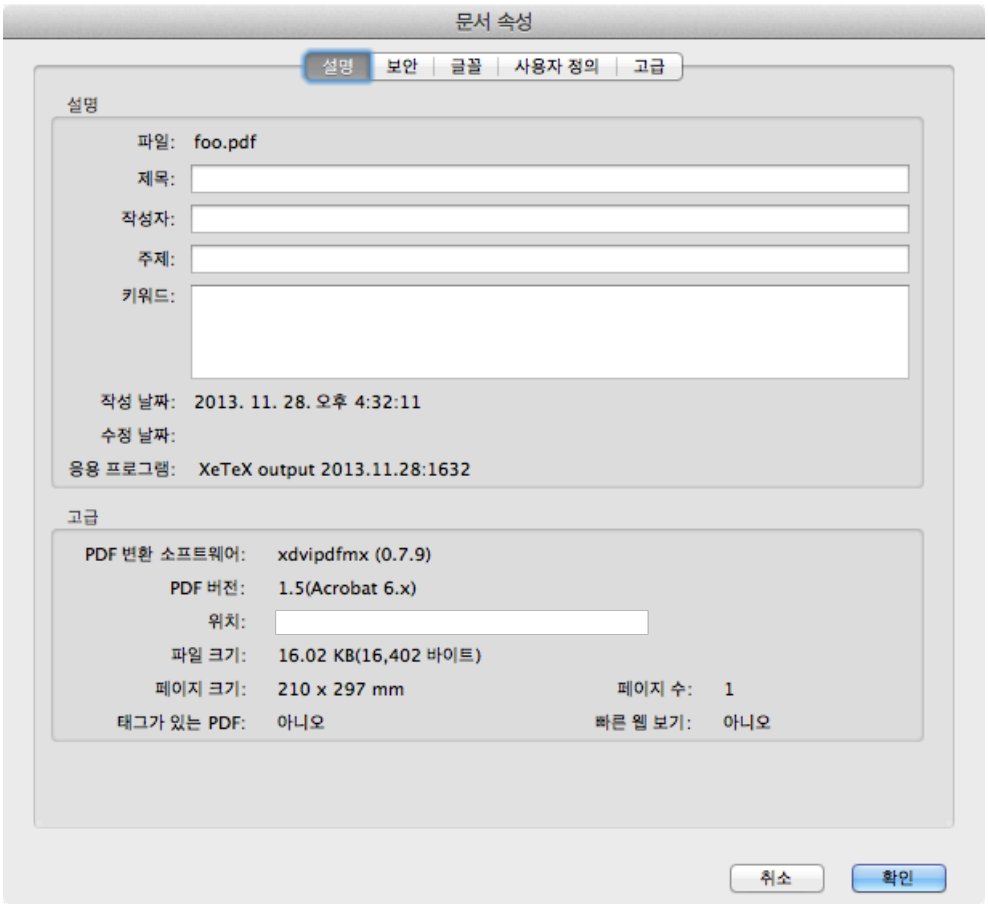
TeX은 강력한 조판시스템이다. 강력한 만큼 배울 게 많다. 앞의 예제코드는 매우 간단한 코드이지만 그 속에는 많은 사연이 들어있다. 본격적으로 시작하기 전에 예제코드를 꼼꼼히 살펴보겠다.

1.4.1 클래스(documentclass)

예제코드의 `\documentclass[oneside]{article}`부분이다. 클래스는 문서의 성격을 결정하는 가장 중요한 부분으로 tex문서를 작성할 때 반드시 가장 먼저 한 번만 쓰인다. 템플릿처럼 문서 전체의 모양을 결정할 뿐만 아니라, 많은 기능과 옵션을 제공한다. 예컨대 예제코드의 `\section{시작하기}` 명령은 article

[7] http://skim-app.sourceforge.net/

그림 1.4 Adobe Reader로 본 문서 속성



클래스가 제공하기 때문에 사용할 수 있다. 하지만 `\chapter{시작하기}`명령을 사용하면 오류가 발생한다. article 클래스에는 없는 명령이기 때문이다. 또다른 표준클래스인 report 클래스는 `\chapter{}`명령을 제공하기 때문에 사용할 수 있다.

oneside 옵션은 article 클래스가 제공하는 옵션이다. 이 외에 a4paper, 11pt, landscape 등의 옵션들이 있다. 어떤 옵션을 쓸 수 있는지는 전적으로 클래스에 달려있다. 만약 클래스에서 제공하지 않는 기능들을 사용하고 싶다면, 관련 패키지를 사용하거나 직접 정의(7.1장 사용자 정의 명령)해야 한다.

1.4.2 명령 (command)

명령은 대체로 다음과 같은 형식을 갖는다.

Code 5

```
\section[시작]{\LaTeX 시작하기}
```

대괄호([])는 옵션이고, 중괄호({})는 명령의 이름이다. 명령에 따라 옵션이 있을 수도 있고 없을 수도 있다. 또 옵션이나 인자가 2개 이상인 명령도 있다. 만약 텍 문서를 컴파일 하는 중에 `! Undefined control sequence.` 오류가 발생한다면, 명령에 오타가 있거나 정의되지 않은 명령(클래스나 사용하는 패키지에 없는 명령)을 쓴 것이다. 이럴 때는 해당 명령을 제공하는 클래스나 패키지 매뉴얼을 읽어봐야 한다. 매뉴얼에 대한 설명은 1.5절 texdoc을 참조하기 바란다.

1.4.3 환경 (environment)

환경은 `\begin{center}`⋯`\end{center}`와 같은 형식을 갖는다. 옵션의 위치는 환경에 따라 조금씩 다를 수 있다.

`\begin{document}`⋯`\end{document}`도 환경이다. `document` 환경은 LaTeX문서라면 반드시 하나만 존재해야 하는 중요한 환경이다. 마치 HTML의 `<body>` 태그와 같다. 환경 역시 정확한 사용방법은 매뉴얼을 보는 것이다.

1.4.4 프리앰블 (Preamble)

프리앰블[8]은 `\documentclass{}`와 `\begin{document}` 사이에 있는 부분을 말한다. 예제코드에서

[8] 전처리부라고 부를 수도 있지만, 나는 프리앰블이 익숙하다.

Code 6

```
\usepackage{kotex}
\setmainhangulfont{HCR Batang LVT}
```

가 위치한 곳이 바로 프리앰블이다. HTML의 <head>태그와 비슷한 기능을 하지만 프리앰블이라고 명시하는 명령은 따로 없다. 패키지를 불러오거나 (\usepackage{}), 명령이나 환경을 정의할 때에는 프리앰블에 해야 한다.

1.4.5 패키지 (package)

패키지를 사용하면 클래스가 제공하지 않는 많은 기능들을 이용할 수 있다. 대표적으로 한국어를 조판하기 위해서는 ko.TEX패키지를 사용해야 한다. 패키지를 사용하는 방법은 프리앰블에서 \usepackage[옵션]{패키지이름}과 같은 방식으로 사용한다. 간혹 패키지끼리 충돌을 일으키거나 읽어들이는 순서 (\usepackage{}명령을 선언한 순서)에 따라 문제가 생길 수 있다. 이런 경우에는 **알아서 잘**(5장 Minimal Working Example) 해결하라.

1.4.6 공백문자와 문단

공백문자란 스페이스, 탭 등을 가리킨다. TEX은 이런 공백문자들을 연속으로 입력해도 하나의 스페이스로 처리한다. HTML과 같다. **한 번의 줄바꿈**(linebreak) 역시 하나의 스페이스로 처리한다.

LATEX으로 문서를 작성할 때 주의해야 할 점은 문장과 문단을 엄격히 구분한다는 것이다. 워드프로세서에서는 문장, 줄바꿈, 문단 등이 아무 개념없이 사용되어도 문제를 느끼지 못하지만 LATEX에서는 다르다. 문단을 나누기 위해서는 문장 끝에 빈 줄을 삽입하면 된다. 또는 \par라는 명령을 사용해도 된다.

Code 7

```
이것은 첫 번째 문단의 첫 번째 문장입니다.    두 번째 문장입니다.



두   번째 문단의 첫 번째 문장입니다. 공백을 많이 쓴다고
공백이 그만큼 생기지 않습니다. 나쁜 습관입니다.
```

결과

이것은 첫 번째 문단의 첫 번째 문장입니다. 두 번째 문장입니다.

두 번째 문단의 첫 번째 문장입니다. 공백을 많이 쓴다고 공백이 그만큼 생기지 않습니다. 나쁜 습관입니다.

1.4.7 특수문자

LaTeX에서는 몇 가지 문자들을 특별한 용도로 사용한다. 따라서 문서 내에 그냥 입력하게 되면 오류가 발생하거나 의도하지 않은 결과가 나올 것이다.

Code 8

```
# $ % ^ _ & \ ~ { }
```

위 특수문자들을 입력하기 위해서는 사용하고 싶은 특수문자 앞에 백슬래시(\)를 더해주면 된다. 다만 \의 경우 \\는 줄바꿈명령이므로, `\textbackslash` 명령을 사용한다.

%는 주석(comment)을 나타내는 특수문자이다. 따라서 LaTeX이 컴파일 할 때 무시해 버린다. HTML의 '`<!-- -->`', Python의 '`#`'와 같다.

#은 사용자 정의 명령이나 환경을 정의할 때 사용되는 기호이다. 7.1.2절 옵션이 있는 명령을 참고한다.

$은 수식을 조판할 때 쓰인다. $와 $ 사이에 수식을 넣으면 LaTeX이 수식을 예쁘게 조판해 준다.

Code 9

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

결과

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

하지만 수식을 입력할 때는 $ 기호 대신 \[, \]를 사용하는 것을 권한다.

Code 10

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

^, _ 은 각각 수식에서 윗첨자, 아래첨자를 뜻한다. 이외 수식과 관련한 내용은 lshort-kr문서의 '제 3장. 수식 조판하기'를 참조하기 바란다.

&는 표(Table)에서 다음 열로 넘어갈 때 쓰인다.

\\는 문단을 나눌 때도 쓰이지만, 표에서는 새로운 행을 추가할 때도 쓰인다.

~(틸데)는 줄바꿈이 안되는, 폭이 고정된 공백을 만들어준다. 만약 ∼을 표시하고 싶은 거라면 sim를 입력한다.

{와 }는 굳이 설명하지 않겠다. \begin{document}만 봐도 어디에 쓰이는지 알 수 있을 것이다.

1.4.8 따옴표

작은따옴표나 큰따옴표를 입력해야 하는 경우가 많다. 이 때 여는 따옴표와 닫는 따옴표의 모양이 조금 다르다. LaTeX에서는 다음과 같이 입력하면 이들 따옴표를 이쁜 따옴표[9]로 만들어 준다.

Code 11

```
`작은 따옴표', ``큰 따옴표''
```

결과

‘작은 따옴표’, “큰 따옴표”

이것이 가능하려면 fontspec이 제공하는 다음 명령을 프리앰블에 선언해야 한다.[10]

Code 12

```
\defaultfontfeatures{Ligatures=TeX}
```

여는 따옴표(grave accent, `)는 키보드에서 숫자 1 왼쪽에 있는 키(틸데(~)와 같은 자리)이고, 닫는 따옴표(vertical quote, ')는 키보드에서 엔터키 왼쪽에 있는 키("와 같은 자리)이다.[11]

1.5 texdoc

texdoc명령을 이용하면 설명서를 쉽게 열 수 있다. 대부분의 클래스나 패키지의 설명서는 <texdoc 패키지이름>과 같은 방식으로 읽을 수 있다. 예를 들어 ko.TeX에 대한 설명서를 읽기 위해서는 터미널 또는 명령프롬프트에서 texdoc

[9] curly quotation mark

[10] 자세한 내용은 KTUG QnA 게시판에 두텁님이 쓰신 답글을 읽어보자.

[11] 이 배열은 QWERTY 자판을 기준으로 한 설명이다.

`kotex` 명령을 실행하면 된다. 그러면 시스템의 기본 PDF 뷰어가 실행되면서 kotexdoc.pdf 파일이 열린다.

제 2 장

기본 명령과 환경

2.1 문서제목, 저자, 날짜

문서를 시작할 때 문서의 제목과 저자 이름을 밝히는 것은 기본이다. 날짜도 입력하면 좋겠다. 프리앰블에 다음을 입력한다.

Code 13

```
\title{근본없는 \LaTeX 설명서}
\author{Dokenzy}
\date{단기 4346년}
```

만약 날짜부분을 \date{\today}와 같이 입력하면 자동으로 오늘 날짜를 입력해준다. 그리고 나서

Code 14

```
\begin{document}
\maketitle
```

와 같이 \begin{document}다음에 \maketitle 명령을 사용하면 된다.

2.2 짜임새있는 문서 만들기

LaTeX은 몇 가지 장절명령을 제공함으로써 논리적인 문서를 만드는 것을 도와준다. \chapter{}는 장(章), \section{}은 절(節), \subsection{}은 소절(小節)을 의미한다. 이 명령들은 문서를 논리적으로 구분해 줄 뿐만 아니라, \tableofcontents 명령(2.4절 차례)이 차례를 만들 때 단락제목, 장절번호, 페이지번호 등을 자동으로 넣어준다. 만약 장절 명령을 쓰고 싶되, 차례에는 나타내고 싶지 않다면 \chapter*{}와 같이 *를 붙여주면 된다.

단락제목이 너무 길면 차례에서 볼 때 불편할 수 있다. 이 때는 \chapter[short title]{very long chapte
과 같이 옵션을 주면, 차례에는 'short title'이, 본문에는 'very long chapter title'이 들어간다.

2.3 목록

문서를 작성하다보면 목록을 만들 필요가 있다. 이런 경우들이다.

Code 15

```
\begin{itemize}
\item 좋아하는 연예인
\item 좋아하는 텍스트 편집기
\item 우리동네에 있는 카페
\end{itemize}
```

결과

- 좋아하는 연예인 이름
- 좋아하는 편집기
- 우리동네에 있는 카페

HTML의 태그와 같다. 항목 기호를 점 대신 하이픈(-)으로 하고 싶다면, 옵션을 넣어준다. 만약 특정 항목의 기호만 바꾸고 싶다면 해당 항목에만 옵션을 주면 된다.

Code 16

```
\begin{itemize}[-]
\item 좋아하는 연예인
\item[*] 좋아하는 텍스트 편집기
\item 우리동네에 있는 카페
\end{itemize}
```

결과

- 좋아하는 연예인

* 좋아하는 텍스트 편집기

- 우리동네에 있는 카페

만약 순서가 중요한 경우라면 itemize 환경 대신 enumerate 환경을 이용하자. TEX이 알아서 번호를 매겨준다. HTML의 태그와 같다.

Code 17

```
\begin{enumerate}
\item 좋아하는 연예인 이름
\item 좋아하는 편집기
\item 우리동네에 있는 카페
\end{enumerate}
```

결과

1. 좋아하는 연예인 이름
2. 좋아하는 편집기
3. 우리동네에 있는 카페

항목 기호를 아라비아 숫자 대신 알파벳 소문자로 나타내고 싶다면…? 그 방법은 이 문서에서 다루지 않는다. 구글에서 검색하기 바란다. 검색 키워드는 'latex enumerate numbering style'.

2.4 차례

차례없는 문서는 핸들없는 자동차와 같다. LaTeX으로 차례를 만드는 방법은 아주 간단하다. 2.1절 문서제목, 저자, 날짜에서 설명한 `\maketitle` 다음에 `\tableofcontents`명령을 사용하면 된다.

Code 18

```
\begin{document}
\maketitle
\tableofcontents
```

중요한 점은 최소 두 번 이상 컴파일해야 차례를 만들 수 있다. 한 번 컴파일하고 나서 PDF문서를 보면 차례에 아무 것도 나오지 않는 것을 쉽게 알 수 있을 것이다. 대신 `foo.tex` 파일과 같은 폴더에 `foo.toc` 파일이 생겼을 것이다. 왜 두 번 이상 컴파일 해야 하는지, 어떻게 자동으로 만들어지는지 궁금한 사람은 foo.toc 파일을 살펴보기 바란다.

장, 절에 대한 차례 뿐만 아니라 그림이나 표의 차례도 자동으로 만들 수 있다. 각각 다음 명령을 사용하자: `\listoffigures`, `listoftables`

2.5 찾아보기 (색인)

큰 문서의 맨 끝에는 보통 찾아보기가 온다. 찾고자 하는 내용이 어디있는지 쉽게 찾아갈 수 있는 기능을 제공한다. LaTeX은 찾아보기도 쉽게 만들 수 있다. 이 문서에서는 XeLaTeX에서 찾아보기를 만드는 방법만 제공한다.

먼저 프리앰블에 다음 명령을 입력한다.

Code 19

```
\usepackage{makeidx}
\makeindex
```

그리고 본문에서 찾아보기에 들어갈 단어를 다음과 같이 입력한다.

Code 20

```
우쭈쭈\index{우쭈쭈}
```

이제 찾아보기에서 우쭈쭈 항목 옆에 쪽번호를 누르면 바로 여기 '우쭈쭈'로 오게 된다.

마지막으로 문서에서 찾아보기가 들어갈 곳에 `\printindex` 명령을 사용한다.

이제 문서를 컴파일해서 idx 파일을 얻은 후, texindy라는 별도의 유틸리티를 사용한 후, 다시 컴파일한다. 다음과 같이 하면 된다.

Code 21

```
# xelatex foo
# texindy -L korean -I omega foo.idx
# xelatex foo
```

만약 차례, 상호참조 등을 사용한다면 xelatex으로 컴파일을 두 번 이상 한다. 끝이다. 이제 찾아보기 항목의 'ㅇ' 밑에 '우쭈쭈'와 쪽번호가 보일 것이다.

찾아보기에 대한 자세한 내용은 ko.TEX설명서의 제 8장을 참고하기 바란다.

2.6 상호참조와 자동조사

상호참조는 문서의 특정 부분을 참조하기 위해 쓰인다. 이 문서에서도 상호참조를 많이 사용했다.

Code 22

```
% 여기서는 코드를 보이기 위해 줄을 나누어 입력했다.
% 실제로는 한 문장이므로 한 줄에 입력했다.
나는 \figurename~\ref{fig:foo}\을
\pageref{sec:test}\pagename에 있는
\ref{sec:test}\sectionname에 넣었다.
```

결과

나는 그림 1.3을 4페이지에 있는 1.3절에 넣었다.

복잡해 보이지만 뭔가 멋지지 않은가? 나는 이 상호참조 부분을 강조해서 설명하지 않을 수 없다. `\figurename`[1]은 '그림'을, `\ref{fig:fooproperty}`는 1.4를 출력해준다. `\pageref{sec:test}`명령은 쪽번호를 자동으로 가져온다. 이 번호들을 직접 입력한다고 생각해보라! 그림을 추가하거나 뺄 때마다, 페이지 수가 줄거나 늘 때마다 **그 짓거리**를 다시 해야한다! 워드프로세서로 A4용지 300페이지 분량의 연간보고서를 만든다고 생각해보라! 그래도 워드프로세서가 좋다고 생각한다면 이 문서를 과감히 삭제하기 바란다.[2]

`\ref{}`명령을 이용하면 장, 절, 그림, 표 등에 대한 참조를 TEX이 자동으로 처리해준다. 뿐만 아니다. ko.TEX이 제공하는 자동조사명령을 이용하면 앞에

[1] 이 명령은 LATEX이 제공하는 기능이지만, kotex을 이용하면 한글 또는 한자로 표기할 수 있다.

[2] 지금 당장 워드프로세서를 언인스톨하라는 뜻이 아니다. 무엇이 더 좋은지만 생각해보라.

오는 단어에 따라 조사 '을' 또는 '를'을 알아서 맞춰준다! 즉, 예제코드에서 `ref{sec:test}`의 번호에 따라 조사를 ko.TEX이 알아서 조정해 주는 것이다. ko.TEX이 제공하는 자동조사는 다음의 열두 가지이다.

Code 23

```
\이 \가, \을 \를, \와 \과, \로 \으로, \은 \는, \라 \이라
```

예제:

Code 24

```
% 실제로는 한 줄로 입력했다.
신데렐라\과 백설공주\은 왕자\를 사이에 두고
머리끄덩이\을 잡고 싸웠다더\이라
```

결과

신데렐라와 백설공주는 왕자를 사이에 두고 머리끄덩이를 잡고 싸웠다더라

이제 자동조사기능까지 쓸 수 있다는 것을 알게 되었다. 그래도 워드프로세서를 고집하겠다는 사람이 있다면? 그런 사람은 방금 전 이 문서를 삭제했을테니 자동조사기능을 모를 것이다.

잠시 자동조사 기능으로 얘기가 빠져버렸다. 그 외 ko.TEX이 제공하는 멋진 기능들은 6장 한국어문서 조판하기에 간단히 소개하였다. 다시 상호참조 사용법만 요약하자면,

Code 25

```
\section{안녕하세요}\label{hello}
이번 장은 \pageref{hello}\pagename에 있는
\ref{hello}\sectionname입니다.
```

참조에 쓰이기 위한 라벨 `\label{hello}`[3]을 붙여놓고, 문서 어딘가에서 `\ref{hello}`, `\pageref{hello}` 등의 명령으로 라벨이 붙은 곳의 정보를 자동으로 가져올 수 있다.

문서의 규모가 크다면 라벨을 달 때 `\label{sec:hello}`와 같이 사용할 수 있다. `sec:`이 특별한 기능을 하는 것은 아니다. 다만 절 (section) 에 대한 라벨임을 쉽게 알 수 있도록 해줄 뿐이다. 물론 이를 참조하기 위해서는 `\ref{sec:hello}`와 같이 해야 한다.

이 편리한 기능을 쓰기 위해서 당신이 해야할 일은 컴파일을 3번 이상 해주는 것 뿐이다. 만약 상호참조가 나와야 할 부분이 '?' 로 나온다면 참조를 잘못했거나, 컴파일을 더 해야 한다. 상호참조에 문제가 있다면 컴파일이 끝난 후 다음과 같은 경고가 나타나니 걱정할 것 없다.

Code 26

```
LaTeX Warning: There were undefined references.



LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right.
```

2.7 글꼴 변경하기

2.7.1 글꼴

LaTeX은 문서의 논리적 기능에 따라 알아서 글꼴 및 글꼴크기를 선택해서 조판한다. 이 매뉴얼은 장제목, 절제목, 본문, 각주 등의 크기가 서로 다르게 조판되었다. 하지만 이것은 oblivoir 클래스 (6.2절 oblivoir 클래스) 가 알아서 해준 것이지, 내가 일일이 지정해 준 것이 아니다. 물론 내 마음대로 바꾸는 것도 당연히 가능하다. 이 문서에서는 본문 글자에 대해서만 다루기로 한다.

[3] XeLaTeX과 LuaLaTeX에서는 라벨에 한글을 쓸 수 있다.

본문에서 글자 모양을 변경하고 싶은 글자가 있다면 아래 명령들을 사용하면 된다.

`\textrm {...}`	roman
`\textsf {...}`	sans serif
`\texttt {...}`	`typewriter`
`\textmd {...}`	medium
`\textbf {...}`	**bold face**
`\textup {...}`	upright
`\textit {...}`	*italic*
`\textsl {...}`	*slanted*
`\textsc {...}`	Small Caps
`\emph {...}`	*emphasized*
`\textnormal {...}`	document font

2.7.2 글자 크기

LaTeX에서 글자 크기를 변경하고 싶은 경우에는 문맥에 따라 상대적인 크기로 정하는 것이 좋다. 다음은 LaTeX에서 글자크기를 변경하는 방법이다.

`\tiny`	제일 작은 크기
`\scriptsize`	scriptsize
`\footnotesize`	각주 크기
`\small`	small
`\normalsize`	본문 기본 크기
`\large`	large
`\Large`	Large
`\LARGE`	LARGE
`\huge`	huge
`\Huge`	제일 큰 크기

이런 글자크기 명령은 이후의 글자들에 계속 적용이 된다. 그래서 중괄호로

감싸서 특정 부분에만 적용할 수 있도록 하자.

Code 27

```
이렇게 하면 {\tiny 중괄호 내에만 영향}을 미친다.
```

결과

이렇게 하면 중괄호 내에만 영향을 미친다.

만약 문단 전체에 적용하고 싶다면 명령대신 환경을 이용하자.

Code 28

```
\begin{footnotesize}
문단 전체 글씨가 각주 사이즈로 나타납니다.
문단 전체 글씨가 각주 사이즈로 나타납니다.
문단 전체 글씨가 각주 사이즈로 나타납니다.
\end{footnotesize}
```

결과

문단 전체 글씨가 각주 사이즈로 나타납니다. 문단 전체 글씨가 각주 사이즈로 나타납니다.
문단 전체 글씨가 각주 사이즈로 나타납니다.

2.8 각주 (footnote)

각주를 달기 위해서는 각주를 달고 싶은 낱말 또는 문장 뒤에[4]

Code 29

```
\footnote{각주는 여기에 나타납니다.}
```

를 입력한다. 각주 역시 상호참조와 번호 매기기가 자동으로 된다. 끝.

[4] 각주는 여기에 나타납니다.

제 3 장

필수 패키지

LaTeX이 제공하는 명령과 환경만으로도 그럴듯한 문서를 만들 수 있다. 하지만 글자 색상을 바꾸거나 무언가 꾸며줄 만한 요소들을 추가해야 할 때도 있다. 이 장에서는 LaTeX으로 문서를 작성할 때 많이 사용하는 패키지들을 소개하고 간단한 사용법을 설명한다.

3.1 graphicx

이미지를 넣기 위해 쓰이는 패키지이다.

Code 30

```
\includegraphics{KTSmeta}
```

결과

이미지 파일이름은 확장자를 빼고 파일 이름만 넣도록 한다.

하지만 이미지만 달랑 넣기 보다는 figure 환경 안에 사용하기를 권한다.

Code 31

```
\begin{figure}[!hbp]
\centering
\caption{KTS META(Duane Bibby)}
\label{fig:ktsmeta}
\includegraphics{KTSmeta}
\end{figure}
```

결과

그림 3.1 KTS META(Duane Bibby)

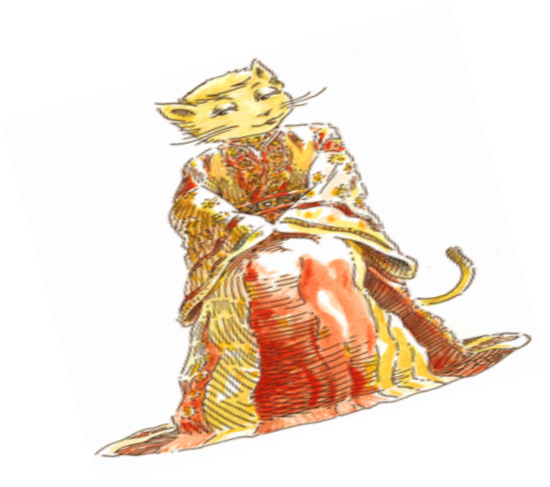
이 명령은 다양한 옵션을 제공하는데 width는 가로, height는 높이, angle은 회전 각도를 뜻한다. 다른 옵션들은 설명서를 참고하라.

여기서 주의할 점은 \label{}명령을 사용할 경우, \caption{}명령 뒤에 사용해야 한다. 그래야 어딘가에서 \ref{}명령으로 내부 참조를 할 때 해당 그림으로 이동할 수 있다.

Code 32

```
\centering
\includegraphics[
        width=0.5\textwidth,
        height=5cm,
        angle=15]{KTSmeta}
```

결과



여기서 `0.5\textwidth`는 본문폭(textwidth)의 절반(0.5)을 뜻한다.

3.2 xcolor

xcolor 패키지는 색을 다루는 패키지이다. xcolor 패키지가 제공하는 다양한 색상 이름은 물론, 사용자가 직접 색상을 임의로 만들고 이름을 지어줄 수 있다.

Code 33

```
\usepackage[cmyk, dvipsnames, svgnames]{xcolor}
```

xcolor 패키지를 불러올 때 위와 같이 옵션을 줄 수 있다. cmyk는 색상 모델이다. cmyk는 인쇄할 때 사용하는 색상 모델이다. rgb나 HTML 등의 모델을 사용할 수 있다.

dvipsnames나 svgnames는 미리 정의된 색상이름들을 사용하기 위한 옵션이다. 이들 옵션이 제공하는 색상이름을 바로 사용할 수 있다. 어떤 이름들이 어떤

색상인지를 보려면 xcolor 패키지 설명서를 참고하기 바란다. 내가 좋아하는 HotPink는 svgnames가 제공하는 색상이다.

Code 34

```
나는 \textcolor{HotPink}{핫핑크}가 좋아
```

결과

나는 핫핑크가 좋아

이제 직접 색상을 만들고 이름을 짓는 방법을 알아보자. `\usepackage{xcolor}`를 한 다음에 아래와 같이 정의한다.

Code 35

```
\definecolor{SunOrange}{cmyk}{0,.47,.78,0}
```

이 매뉴얼을 여기까지 읽은 사람이라면 이 코드가 어떤 의미인지 알 수 있으리라 생각한다. 다만 마지막에 있는 숫자에 대해서만 간략히 설명하기로 한다. 각 숫자는 Cyan, Magenta, Yellow, Black(Key)에 대한 값이다. RGB가 각각 0부터 255 사이의 값이라면, CMYK는 0부터 1사이의 값이다.

이제 이 색상을 이용해 글자색을 바꿔보자.

Code 36

```
\textcolor{SunOrange}{햇빛오렌지색}
```

결과

햇빛오렌지색

3.3 mdframed

이 놀라운 패키지는 박스를 꾸밀 때 필요한 모든 기능을 제공한다. 지난 2012년 한국텍학회 학술대회에서 이주호님께서 소개하셨는데, 그 때 만드신 발표자료가 정말 훌륭하다. 나중에 이 패키지의 매뉴얼에도 소개되었다[1]. 이 패키지에 대한 설명은 이주호님의 발표자료를 링크하는 것으로 대신하고, 여기서는 이주호님의 예제코드 중 하나만 보이기로 한다.

이 패키지를 사용하기 위해서는 먼저 아래와 같이 usepackage를 해준다.

Code 37

```
\usepackage[framemethod=tikz]{mdframed}
```

TikZ[2]는 프로그래밍 방식으로 그래픽 요소를 만드는 TEX패키지이다. 박스에 멋진 효과를 줄 때 이 TikZ 패키지를 이용해서 만들겠다는 의미이다.

이제 본문에서 mdframed로 박스를 만들어 보자.

[1] http://progress.tistory.com/170
[2] TikZ 홈페이지

Code 38

```
\begin{mdframed}[
        linecolor=Green!80,
        roundcorner=10pt,
        backgroundcolor=Green!10,
        innertopmargin=\topskip,
        frametitleaboveskip=2pt,
        frametitlefont=\bfseries\itshape,
        frametitlerule=true,
        frametitlerulewidth=1pt,
        frametitlebackgroundcolor=Olive!20,
        frametitlealignment=\center,
        leftmargin=1cm,
        rightmargin=1cm,
        frametitle={\normalfont\bfseries\Large%
                \color{DodgerBlue}푸리에 급수의 부분합}]
디리클레는 푸리에 급수의 부분합을 아래와 같이 나타냈습니다.
\[
S_n(x)=\frac{1}{\pi}\int_{\pi}^{\pi}f(t)\frac{sin(n+1/2)(t-x)}{sin(t-x)/2}dt
\]
\end{mdframed}
```

위 코드의 결과는 다음과 같다.[3]

[3]수식코드에 쫄지 않아도 된다. 수학자가 아니라면 이렇게 입력할 일이 없고, 수학자라면 이런 수식은 어렵지 않을 것이다.

결과

푸리에 급수의 부분합

디리클레는 푸리에 급수의 부분합을 아래와 같이 나타냈습니다.

$$S_n(x) = \frac{1}{\pi}\int_{\pi}^{\pi} f(t)\frac{sin(n+1/2)(t-x)}{sin(t-x)/2}dt$$

3.4 hyperref

문서를 작성하다보면 링크를 만들 경우가 생긴다. 문서 외부의 자료를 참조하는 링크 뿐만 아니라 문서 내에서 참조하는 내부 참조 역시 링크이다. hyperref 패키지는 이런 링크들을 다룬다.

대표적인 명령으로 이런 명령이 있다.

Code 39

```
\href{http://www.ktug.org}{KTUG}
```

어떻게 사용하는지 쉽게 짐작할 수 있을 것이다. hyperref 패키지는 링크의 색상도 지정할 수 있다. 또한 PDF문서의 속성까지도 지정할 수 있다.

다음은 이 문서에서 사용한 hyperref 패키지 옵션이다.

Code 40

```
\usepackage[
    pdfauthor={dokenzy},
    pdftitle={모두를 위한 LaTeX},
    pdfsubject={LaTeX을 시작하려는 사람들을 위한 성의없는 설명서},
    pdfkeywords={LaTeX, 매뉴얼, 너무 성의없다},
    colorlinks= true,
    linkcolor=blue,
    urlcolor=blue,
    citecolor=blue,
    anchorcolor=blue]{hyperref}
```

그런데 코드 상에는 아무 문제가 없는데 컴파일 할 때 '! LaTeX Error: Option clash for package hyperref.' 오류가 발생할 수 있다. 사용중인 어느 패키지가 이미 내부적으로 hyperref 패키지를 사용하고 있기 때문이다.[4] 이럴 때는 hyperref 패키지의 옵션만 설정하자. 프리앰블에서 `\hypersetup{}` 명령으로 설정한다.

[4] 어떤 패키지가 다른 패키지를 로드할 때는 보통 `\usepackage{}`대신 `RequirePackage{}` 명령을 사용한다.

Code 41

```
\hypersetup{
    pdfauthor={dokenzy},
    pdftitle={모두를 위한 LaTeX},
    pdfsubject={LaTeX을 시작하려는 사람들을 위한 성의없는 설명서},
    pdfkeywords={LaTeX, 매뉴얼, 너무 성의없다},
    colorlinks= true,
    linkcolor=blue,
    urlcolor=blue,
    citecolor=blue,
    anchorcolor=blue}
```

3.5 tabu

LaTeX에서 표를 만드는 것은 손이 많이 가는 작업이다. tabu는 기존의 불편함을 많이 해소하고 괜찮은 인터페이스와 강력한 기능을 제공한다. 이주호님은 2011년 한국텍학회 학술대회에서 '표 만들기의 새 표준 tabu'라는 제목으로 이 패키지를 소개해 주셨다. 이 때의 발표자료 역시 아주 좋은 참고자료이다.

이주호님의 자료가 정말 좋기 때문에 여기서는 이주호님의 발표자료 중 하나만 예를 보이는 것으로 마치겠다.

Code 42

```
\begin{tabu} spread 10pt {X[c]*{3}{|X[2,1]}}
\toprule
$x$ (도) & $\sin x$ & $\cos x$ & $\tan x$ \\
\midrule
\rowfont[c]{\sffamily} 0 & 0 & 1 & 0 \\
1 & 0.0174524 & 0.99984769 & 0.01745506 \\
\rowfont[r]{\large\itshape\bfseries}
2 & 0.03489949 & 0.99939082 & 0.03492076 \\
3 & 0.05233595 & 0.99862953 & 0.05240777 \\
\bottomrule
\end{tabu}
```

결과

x (도)	$\sin x$	$\cos x$	$\tan x$
0	0	1	0
1	0.0174524	0.99984769	0.01745506
2	***0.03489949***	***0.99939082***	***0.03492076***
3	0.05233595	0.99862953	0.05240777

제 4 장

폰트

XeLaTeX은 시스템에 설치된 트루타입과 오픈타입 폰트들을 직접 사용할 수 있다. Fontspec 패키지는 이 폰트들을 직관적인 방법으로 사용할 수 있게 해준다. ko.TeX은 내부적으로 Fontspec 패키지를 이용한다. 이주호님이 AJT[1]에 쓰신 Fontspec: XeTeX의 날개라는 글은 XeTeX과 Fontspec 뿐만 아니라 폰트 기술 전반에 대해 정리한 귀한 글이다.[2]

다양한 폰트들을 보여주려는 특별한 목적이 아니라면 하나의 문서에 3개의 폰트패밀리 (font family) 를 사용하는 것을 권장한다. 바탕체 (serif), 돋움체 (sans serif), 타자체 (typewriter) 가 바로 그것이다. 바탕체는 본문에 쓰이는 기본 폰트, 돋움체는 큰 제목이나 캡션 등에 쓰이는 폰트, 타자체는 그 외 특별한 용도에 쓰인다고 보면 된다.

이들 폰트를 지정하는 방법은 다음과 같다.

[1] the Asian Journal of TeX

[2] 나는 폰트 기술을 다룬 문서 중에 이보다 좋은 한글문서를 찾지 못했다.

Code 43

```
\setmainfont{serif}
\setsansfont{sanserif}
\setmonofont{typewriter}
\setmainhangulfont{바탕글꼴}
\setsanshangulfont{돋움글꼴}
\setmonohangulfont{타자글꼴}
```

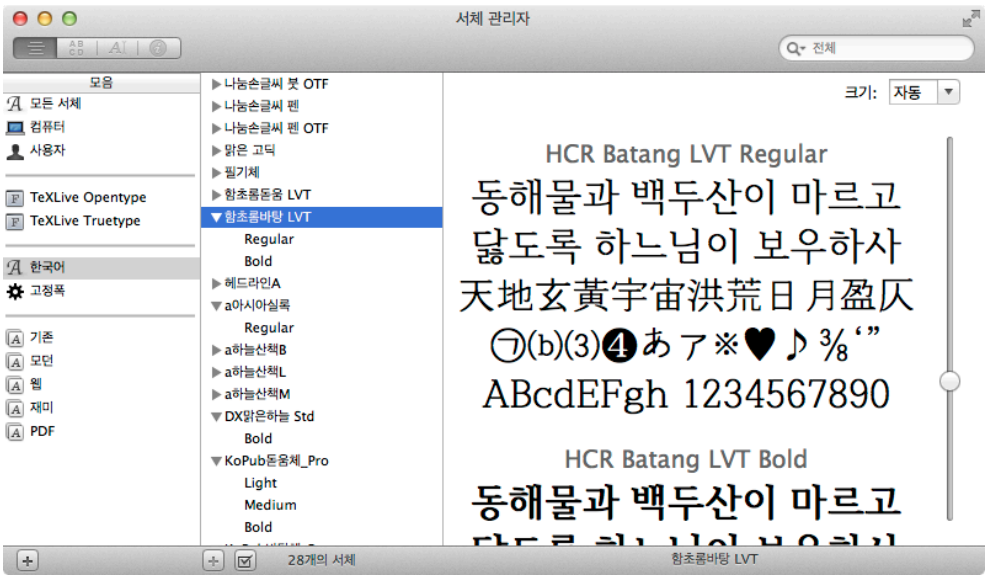
예컨대, 이 문서에 쓰인 한글본문폰트는 다음과 같이 설정했다.

Code 44

```
\setmainhangulfont{HCR Batang LVT}
```

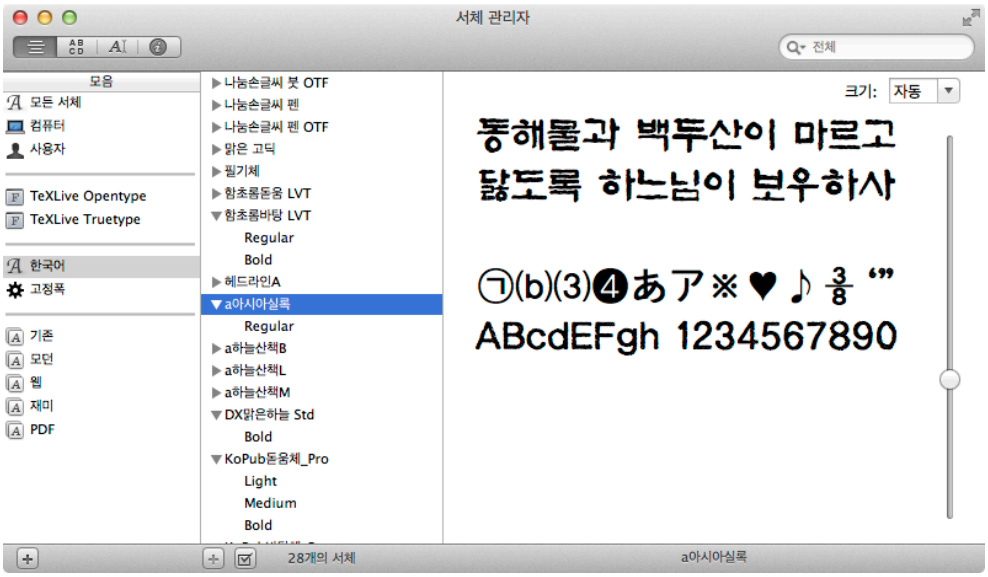
HCR Batang LVT는 절 한국텍학회와 한글텍사용자그룹에서 언급한 폰트이다. 이 폰트의 이름은 폰트유틸리티 등을 통해 알 수 있다(그림 4.1).

그림 4.1 서체관리자로 본 함초롬바탕 LVT 폰트



하지만 그림 4.2처럼 이름이 없는 경우도 있다.

그림 4.2 서체관리자로 본 a아시아실록 폰트



이런 경우에는 다음과 같이 폰트의 파일명으로 지정할 수 있다.[3]

Code 45

```
\setsanshangulfont{a아시아실록.otf}
```

영문폰트와 한글폰트를 따로 지정하는 이유는, 한글폰트에 들어있는 영문폰트가 영어권 회사의 폰트보다 좋지 않기 때문이다. 따라서 한글폰트와 잘 어울리는 영문폰트를 선택하는 것이 중요하다. 무슨 폰트를 써야할지 막막한 사람은 이주호님이 문서작성워크숍에서 발표한 자료(타이포그래피)를 참고하자.

[3] 폰트파일명이 한글인 경우 안될 수 있다. 그럴 때는 파일명을 영문자로 바꾸자. 자세한 내용은 `oblivoir-simpledoc.pdf`의 5.5절 ExternalLacation을 살펴보라.

제 5 장

Minimal Working Example

이제 LaTeX으로 많은 것들을 할 수 있을 것이다. 차례, 상호참조, 목록, 박스꾸미기, 표, … 하지만 컴파일하다가 오류를 만난 확률도 그만큼 높아질 것이다. 이제 어떻게 하면 오류를 찾고 해결할 수 있는지 그 방법을 찾아보자.

대부분의 오류는 다음과 같은 경우에 발생한다.

- 오타
- 괄호의 짝이 맞지 않을 때
- `\begin{}`, `\end{}` 짝이 맞지 않을 때
- 없는 명령을 사용했을 때
- 없는 환경을 사용했을 때
- 옵션을 잘못 사용했을 때

단순한 명령과 환경만으로 만든 텍 문서에서 오류를 해결하는 방법은 그리 어렵지 않다. 하지만 많은 패키지를 쓰고 직접 명령이나 환경을 정의해서 사용하는

큰 규모의 문서라면 오류를 찾는 것 자체가 어려울 것이다. 다행히 LaTeX은 컴파일시 발생하는 오류에 대해 꽤 정확하게 알려준다. 컴파일 중에 오류가 발생하면 먼저 오류 메시지를 잘 읽어보자.

오류메시지를 잘 읽어보면 어느 부분에서 오류가 발생했는지 알 수 있다. 이 오류메시지가 아주 정확한 경우도 있지만 그렇지 않은 경우도 있다. 어떤 부분의 오류 때문에 다른 부분이 오류가 발생하는 경우이다. 이런 오류는 원인을 알아내는 것이 어려울 수 있다. 패키지가 서로 충돌하는 경우도 어렵다. 하지만 Minimal Working Example(MWE)을 만들어보면 이런 경우도 오류의 원인을 비교적 쉽게 찾아낼 수 있다. 이 장의 목표는 오류의 원인을 찾는 것이지 오류를 해결하는 방법을 찾는 것이 아니다.

5.1 왜 MWE를 만들어야 하는가?

문서를 만들다보면 이런저런 명령과 환경을 쓰기 위해 패키지를 사용하기 마련이다. 그러다가 에러메시지만으로 오류의 원인을 찾을 수 없게 되면 곤란해진다.[1] MWE는 바로 그런 경우에 특히 유용하다. 먼저 현재의 파일을 복사(`foo-mwe.tex`)한다. 복사한 파일에서 오류와 관련없는 부분을 조금씩 삭제하면서 컴파일을 해본다. 이렇게 하다보면 원래 문서에서 발생한 오류과 같은 오류를 갖는 최소한의 텍 코드를 만들게 된다. 비록 컴파일할 때 오류가 발생하지만 어떤 환경(엔진, 클래스, 패키지, 명령, 인코딩 등 많은 조건들)에서 그런 오류가 발생하는지 알 수 있다.

이 문서를 작성하면서 의도한대로 결과가 나오지 않아 몇 번의 MWE를 만들어야 했다. 가장 최근에 만든 MWE 작업과정을 예로 들어보이겠다.

그림의 캡션 모양을 수정하기 위해 프리앰블에 다음과 같은 코드를 입력했다. 하지만 결과가 바뀌지 않았다.

[1] 제일 곤란한 것은 오류없이 컴파일은 되는데 원하지 않은 결과가 나오는 것이다.

Code 46

```
\captiondelim{ }
\captionnamefont{\footnotesize\bfseries\sffamily}
\captiontitlefont{\footnotesize\sffamily}
```

50페이지 분량의 컴파일 과정이 후다닥 지나가 버린다. log파일도 2500줄이 넘는다. MWE를 만들어야겠다.

1. latex4all.tex파일을 captiontest.tex파일로 복사한다.
2. 그림 캡션이 의도한대로 잘 나오는지 확인만 하면 되니까 첫번째 그림부분을 제외하고 본문에서 내용을 전부 삭제한다[2].
3. 프리앰블에서도 그림과 상관없는 것들을 모두 지운다.
4. 이제 의도한대로 잘 나온다. 프리앰블에 뭔가 잘못된 것이 있을 것이다.
5. 그림 캡션과 관련이 있을테니 이와 관련된 패키지를 찾아봐야겠다.
6. \usepackage{caption}이 있다. 이걸 다시 넣고 테스트해보자.

이런 경고가 보인다.

Code 47

```
Class memoir Warning: You are using the caption package with
the memoir class. To prepare we will now reset all captioning
macros and configurations to kernel defaults, and then let the
caption package take over. Please remember to use the caption
package interfaces in order to configure your captions.
```

[2]당연히 문서 맨 끝에 \end{document}는 있어야한다.

그렇다. 내가 사용하는 oblivoir라는 클래스는 memoir클래스 위에서 동작하는데, memoir클래스에서 제공하는 캡션 관련 명령들이 caption 패키지 때문에 동작하지 않은 것이었다.

물론 이 경고 메시지는 1,000줄 짜리 latex4all.tex파일을 컴파일 할 때도 있었다. 하지만 2,500줄이 넘는 log파일에서 이걸 찾아내려면…

이제 원인을 알았으므로 문제를 해결해보자. 다행히 이 문제는 \usepackage{caption}을 지우는 것으로 해결할 수 있었다. 하지만 어떤 경우엔 해결방법을 찾는 것이 어려울 때도 있는 법, 이럴 때 해결하는 방법은 1) 구글이나 KTUG에서 검색하거나, 2) KTUG에 질문하는 것이다.

검색은 직접 해보면 되고, 이 문서에서는 KTUG에 질문을 올리는 요령을 간략히 설명한다.

1. MWE를 만든다.
2. 그림파일이 있다면 그림파일도 포함한다.
3. KTUG의 질문게시판에 자신의 텍 환경(운영체제, TeXLive 버전, 사용한 텍 엔진 등)과 오류내용, 원하는 결과 등을 적고
4. 파일을 첨부해서 올린다. 압축포맷은 zip이 가장 좋다.

'이 정도면 되겠지?'하고 위아래 싹뚝 자르고 코드 일부만 올리고 질문하는 분들이 있다. 또는 아무 코드 없이 얘기만 하는 분들도 있다. 이렇게 올리면 대답도 추상적일 수 밖에 없다. MWE파일을 만들어서 올리자. 이것이 서로 편한 길이다.

텍 코드에는 오류가 없는데도 컴파일 중에 오류가 발생할 수 있다. 텍은 컴파일을 하면서 몇 가지 부수파일들을 만드는데 이런 부수파일 때문에 오류가 발생할 수 있다. TeXworks에서는 [메뉴]-[부수파일 삭제]를 통해 쉽게 지울 수 있다. 웹브라우저에서 캐쉬를 지우는 것과 비슷하다.

5.2 경고(WARNING) 다루기

컴파일을 하면 경고가 발생하는 것을 쉽게 볼 수 있다. 특히 `Overfull` 오류를 자주 볼 수 있다. 이 경고는 내용 일부가 본문 너비보다 넓은 경우에 발생한다. 어떤 경고는 무시해도 되고 어떤 경고는 고쳐야 하는 경우도 있다. 판단은 각자 알아서 잘 한다.

폰트때문에 경고가 발생하는 경우도 있다.[3]

Code 48

```
** WARNING ** Unrecognized OpenType/TrueType cmap format.
** WARNING ** Unable to read OpenType/TrueType Unicode cmap table.
** WARNING ** Failed to load ToUnicode CMap for font "AppleSymbols"
```

대부분 폰트 문제이다. 좋은 폰트는 이런 경고가 없다. 그래서 폰트를 잘 골라 써야 한다.

상호참조에 문제가 있어도 경고가 발생한다. 다음은 이 문서를 컴파일하는 중에 나온 경고이다.

Code 49

```
LaTeX Warning: Reference `chap:xcolor' on page 31 undefined on input line 766.
```

`\label{sec:xcolor}`라고 라벨을 붙였는데, 참조할 때는 `\ref{chap:xcolor}`라고 해서 발생한 경고이다.

코드에 아무런 문제가 없어도 상호참조 경고가 발생할 수 있다.

Code 50

```
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
```

[3] 이 문서를 작성할 때 발생한 경고인데 나는 무시하기로 했다.

이 경고는 텍 파일에 오류는 없지만 참조하는 라벨이 해당 위치에 없는 경우 발생한다. 예를 들면 내용이 변경되면서 쪽번호나 장절번호가 바뀔 수 있는데 이것이 반영이 아직 안됐을 때 발생한다. 해결하는 방법은 한 번 더 컴파일해주면 된다. 만약 해결하지 않으면 쪽번호가 나와야 할 곳이 '?'로 나오게 된다.

제 6 장

ko.TEX과 oblivoir로 한국어문서 조판하기)

3장 필수 패키지까지만 배워도 꽤 멋진 문서를 만들 수 있다. 하지만 한국어 문서를 작성하기 위해서는 여기서 소개하는 ko.TEX을 배울 필요가 있다. ko.TEX은 단순히 한글 입출력에 그치지 않고 '한국어 문서 조판'을 하기 위한 많은 기능들을 제공한다. ko.TEX은 매우 많은 기능을 제공하기 때문에 내가 다 알지 못한다. 여기서는 이런 기능들이 있다는 소개만 간략히 한다.

6.1 ko.TEX

영어권에서 만들어진 TEX을 이용해서 한글을 조판하기 위해서 많은 노력들이 있었고, 그 결과 현재의 ko.TEX v2.0이 만들어졌다. ko.TEX은 다음의 패키지들을 총칭한다.

ko.TEX-utf legacy TeX엔진에 대응하는 패키지

cjk-ko CJK패키지 (개발자 Werner Lemberg) 를 이용하여 유니코드 한글을

식자하는 패키지

XeTeX-ko XeTeX엔진에 대응하는 한글 식자 패키지

LuaTeX-ko LuaTeX엔진에 대응하는 한글 식자 패키지

oblivoir memoir클래스를 이용한 한글 문서 작성 클래스와 패키지 묶음[1]

ko.TeX유틸리티 한글 문서 작성에 필요한 유틸리티들

어려울 수 있지만 어렵지 않게 쓸 수 있다. **우리는 그냥 이렇게만 하면 된다**.

Code 51

```
\usepackage{kotex}
```

위와 같이 선언하면 ko.TeX패키지가 알아서 필요한 패키지를 불러온다. 예를 들어, 텍 엔진이 XeLaTeX이면 XeTeX-ko패키지를 불러온다. 만약 텍 엔진이 LuaLaTeX이라면 LuaTeX-ko패키지를 불러온다. 즉, ko.TeX패키지가 하는 일은 텍 엔진에 따라 필요한 패키지를 자동으로 호출하는 것이다.

ko.TeX이 제공하는 기능들을 나열해보면,

- 우리말 이름(장, 절, 차례, 등)
- 방점, ⋆이런 방점까지!
- 한글식 카운터: ㄴ, ㈀, ㉡, 나, ㉮, ㈏, ②, ⑵, ⓑ, ⒝, 둘, 둘째, ⅱ, Ⅱ
- 자동조사
- 미세조정(자간, 어간, 행간, 행 나눔, 수식과의 간격 등)

[1] 이 문서는 이 클래스로 만들어졌다.

- 한글 책갈피, 찾아보기, 참고문헌 등
- 옛한글
- 한글문서 서식
- 세로쓰기

자세한 내용은 k*o*.TEX설명서를 읽어보기 바란다. 여기서는 세로쓰기의 예만 간략히 보이겠다.

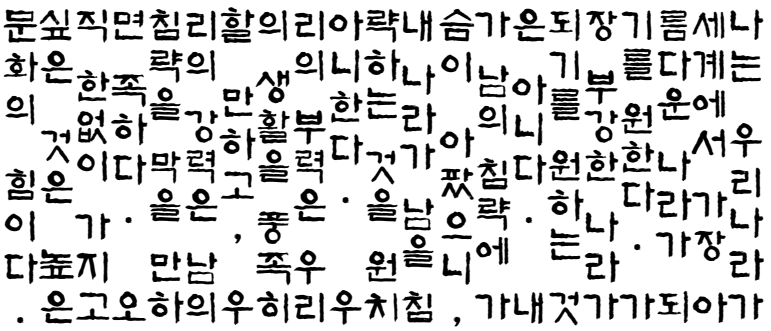


그림 6.1 *

'내가 원하는 우리나라(백범 김구)' 중 일부

6.2 oblivoir 클래스

k*o*.TEX을 이용하면 아름다운 한국어 문서를 조판할 수 있다. 그러기 위해서는 다양한 옵션들을 잘 활용해야 한다. 하지만 나처럼 전문가가 아닌 사람은 원하는 게 무엇인지 조차 잘 모를 수 있다. 옵션을 주어도 제대로 쓸 줄을 모른다.

oblivoir는 바로 그런 사람들이 좋아할 만한 클래스이다. memoir 클래스[2]에 k*o*.TEX을 내장한 이 클래스는 한국어 문서 조판을 위한 설정이 잘 되어 있다. 따라서 k*o*.TEX패키지를 따로 불러들이지 않아도 된다. 이 문서 역시 oblivoir를 사용한다.

[2]많은 기능을 내장한 클래스. 한글로 번역된 문서

따라서 대부분의 한국사람에게 가장 적절한 한국어 문서 제작 방법은 oblivoir 클래스를 이용하는 것이다. oblivoir 역시 좋은 설명서가 있으니 그것을 참고하도록 하자. `<texdoc oblivoir>`

제 7 장

좀 더 LaTeX스럽게

지금까지 LaTeX과 ko.TeX의 기본 명령들을 살펴봤다. 하지만 LaTeX이 강력한 이유는 그 자체로 하나의 언어이기 때문에 사용자가 직접 명령이나 환경을 만들어 쓸 수 있다는 것이다.

7.1 사용자 정의 명령

지금까지는 클래스나 패키지가 제공하는 명령만 사용했다. 이제 직접 명령을 만들어보자.

7.1.1 가장 간단한 명령

만약 C++언어를 만든 비야네 스트롭스트룹(Bjarne Stroustrup)에 대해 글을 쓴다고 하자. 당연히 이 사람의 이름이 자주 쓰일 것이다. 그런데 철자가 길고 어렵다. 뿐만 아니라 영어로 표기할지 한글로 표기할지도 결정하지 못했다. LaTeX에서는 아무 문제가 안된다.

Code 52

```
...
\newcommand{\bjarne}{Bjarne Stroustrup}
...
\begin{document}
\bjarne
...
```

\newcommand 명령은 새 명령을 정의하는 명령이다.

결과

Bjarne Stroustrup

만약에 이름을 한글로 바꾸어야 한다면

Code 53

```
\newcommand{\bjarne}{비야네 스트롭스트룹}
```

으로 선언하면 된다. 이제 본문에서 \bjarne라고 쓴 곳은 전부 '비야네 스트롭스트룹'이라고 나올 것이다.

7.1.2 옵션이 있는 명령

3.2장 xcolor의 예제에서 다음과 같은 명령을 사용했다.

Code 54

```
\textcolor{SunOrange}{햇빛오렌지색}
```

햇빛오렌지색을 자주 사용한다면 좀 더 간편한 방법을 찾아보자.

Code 55

```
\newcommand{\sunge}[1]{\textcolor{SunOrange}{#1}}
\sunge{간편하게 쓰는 햇빛오렌지색}
```

결과

간편하게 쓰는 햇빛오렌지색

7.1.3 옵션에 기본값을 주는 명령

방금 전처럼 하면 항상 햇빛오렌지색만 쓸 수 있다. 만약 다른 색상을 쓰고 싶다면 저런 명령을 색상마다 만들어야 할까? 제일 좋은 방법은 햇빛오렌지색을 기본값으로 주고, 다른 색상을 쓰고 싶을 때만 색상을 지정해 주는 것이다.

Code 56

```
\newcommand{\sunnytext}[2][SunOrange]{\textcolor{#1}{#2}}
기본색은 \sunnytext{햇빛오렌지색}이지만,
다른색도 \sunnytext[Goldenrod]{Goldenrod색}도 할 수 있어요.
```

결과

기본색은 햇빛오렌지색이지만, 다른색도 Goldenrod색도 할 수 있어요.

결과적으로 `\textcolor{}{}`명령과 별반 다르지 않게 됐지만, 이런 식으로 인자와 옵션이 들어가는 명령을 만들 수 있다는 예시로는 괜찮다고 생각한다.

7.1.4 renewcommand

이미 사용중인 명령(그 명령이 클래스에서 제공하든, 패키지가 제공하든, 직접 정의한 것이든)을 다시 정의해야 할 경우가 생기면 `\renewcommand`명령을 사용한다. 이 명령은 사용자 정의 명령보다는 클래스나 패키지가 제공하는 명령을 수정해야 할 경우에 많이 사용된다. 클래스나 패키지를 직접 수정하는 것은 절대

하지 않기를 바란다. 대신 renewcommand를 사용하여 다시 정의하도록 하자. 사용방법은 \newcommand{}와 같다.

7.2 사용자 정의 환경

\newenvironment{}명령으로 사용자 정의 환경을 만들 수 있다.

다음은 lshort-kr문서에 나온 예제이다.

Code 57

```
\newenvironment{king}
        {\rule{1ex}{1ex}%
                \hspace{\stretch{1}}}
        {\hspace{\stretch{1}}%
      \rule{1ex}{1ex}}
\begin{king}
My humble subjects \ldots
\end{king}
```

결과

■ My humble subjects … ■

7.3 파일 나누기

LaTeX의 가장 큰 장점 중 하나는 파일을 분리하여 관리할 수 있다는 점이다. 대규모 문서, 또는 공동으로 문서를 작성할 때 특히 유용하다.

Code 58

```
\include{filename}
```

이 명령은 filename을 현재 tex 파일에 삽입한다. 이 때 호출된 페이지는 새 페이지에서 시작하게 된다.[1]

Code 59

```
\includeonly{file1, file2, file3}
```

이 명령은 프리앰블에서만 쓸 수 있다. `\include{}`명령을 사용할 때 여기에 열거된 파일들만 삽입할 수 있다.

만약 다른 파일을 삽입할 때 새 페이지로 시작하는 것이 싫으면 `\include{filename}` 대신 `\input{filename}` 명령을 사용하자.

[1] 내부적으로 `\clearpage` 명령이 호출되기 때문이다.

제 8 장

추천 문서

내가 이 문서를 쓰기까지 많은 자료를 공부하고 참고했다. 꼼꼼히 읽은 문서도 있고 어려워서 잘 모르고 넘어간 문서도 있다. 한 번 읽고 넘어간 문서도 있고 몇 번이고 본 문서도 있다. '클래스 oblivoir와 책 만들기'같은 내 수준에는 어려운 문서도 있고, Donald Knuth교수의 일화가 실린 재밌는 문서도 있다. 본문에 소개된 문서도 있고 그렇지 않은 문서도 있다. LaTeX을 써보고 싶은 사람이라면 필수 매뉴얼 정도는 정독하길 바란다.

8.1 LaTeX초보자를 위한 필수 매뉴얼

lshort-kr LaTeX전반에 대해 상세하게 다루는 문서

kotexdoc ko.TeX전반에 대해 상세하게 다루는 문서.

xetexko-doc ko.TeX중 XeTeX-ko에 대해 다루는 문서.

8.2 LaTeX을 더 알고 싶은 사람을 위한 글

한국텍학회에서는 'The Asian Journal of TeX'이라는 학술지를 통해 텍을 이용한 조판 및 인쇄 기술 전반에 대해 많은 논문을 발표했다. 여기 실린 모든 글들이 훌륭하지만 ko.TeX및 XeLaTeX과 밀접한 글 몇 가지만 소개를 한다.

클래스 oblivoir와 책 만들기 김강수

한글 텍 : 과거, 현재, 그리고 미래 김강수

Fontspec: XeTeX의 날개 이주호

Oblivoir를 이용한 문서 작성 이기황

8.3 그 외 참고할만한 글

Memucs Manual 김강수 옮김

한없이 다양한 분야에 걸린 관심, 하나의 공통된 맥락 데니스 샤샤, 캐시 레이저어, 박영숙 옮김[1]

레이텍을 빨리 쓸 수 있는 방법 2009 이호재

워드 프로세서 사용자를 위한 LaTeX 김강수 옮김

라텍 소개 및 기본 사용방법 조진환

Introduction to Digital Typography 조진환

XeLaTeX문서작성의 실제 김강수

스타일 작성을 위한 명령어 모음 2008년 공주대 워크숍 강좌팀

[1] 세종연구원에서 출간된 '컴퓨터를 만든 15인의 과학자(Out of their minds)' 중 Donald Knuth 편

article 클래스를 파헤쳐보자 2008년 공주대 워크숍 강좌팀

문단과 리스트 김강수

LaTeX활용하기 김강수

찾아보기

마치며

근본없는 LaTeX매뉴얼은 여기까지다. 힘들어서 맞춤법 수정도 포기했다. tex 소스가 겨우 1050줄에 불과한, A4기준 50쪽도 안되는 문서인데도 불구하고 힘들다. memoir설명서나 lshort-kr같은 문서를 번역하신 분들의 노고가 새삼 더 크게 느껴진다.

영국이나 미국 등에서 초등학생부터 컴퓨터 프로그래밍 교육을 시작한다고 한다. 특히 영국은 5세부터는 간단한 프로그래밍 과정을 가르치고, 11세 이상 학생에게는 실제 프로그래밍 언어를 교육한다고 한다. 나는 LaTeX교육도 프로그래밍 교육 이상의 효과가 있을 것이라고 생각한다. 프로그래밍하듯이 글쓰기를 하니 이보다 좋은 교육방법이 있을까? 또한 웹퍼블리셔에게도 좋은 훈련 방법이 될 것이다. validator만 통과한다고 좋은 마크업이 아니잖은가.

Dokenzy