The Not So Short
Introduction to LaTeX 2ε

LaTeX 2ε 입문

94분 동안 익히는 LaTeX 2ε

by Tobias Oetiker

Hubert Partl, Irene Hyna and Elisabeth Schlegl

Version 3.20, 09 August, 2001

김강수, 강윤배, 장대훈, 김재우

이재승, 현범석, 주철

옮김

3.20-kr, 2002년 3월 31일

감사의 말

이 책은 오스트리아에서 나온 독일어로 쓰여진 LaTeX 2.09 안내 책자로부터 인용한 것이 많다. 저자들은 다음과 같다.

Hubert Partl `<partl@mail.boku.ac.at>`
Zentraler Informatikdienst der Universität für Bodenkultur Wien

Irene Hyna `<Irene.Hyna@bmwf.ac.at>`
Bundesministerium für Wissenschaft und Forschung Wien

Elisabeth Schlegl `<no email>`
in Graz

이 독일어 문서가 보고 싶으면, 아래 사이트에서 Jörg Knappen이 LaTeX 2_ε에 맞게 개선한 버전을 찾아보면 된다.
`CTAN:/tex-archive/info/lshort/german`

이 책을 쓰면서, `comp.text.tex`에 글을 올리는 사람들에게 자문을 구하였는데 많은 분들이 답변을 보내 주셨다. 아래에 기록한 분들은 교정, 제안, 자료들을 보내주셔서 이 글이 더 좋은 글이 되도록 도와주신 분들이다. 이 글이 현재 모양을 갖추는 데 이들의 도움이 컸다. 이 분들 모두에게 진심으로 감사드리고자 한다. 당연히, 이 책에서 잘못된 부분이 있다면 그건 순전히 내 책임이다. 제대로 기록된 것이 있다면 그건 틀림없이 아래 있는 분들이 내게 몇 자 적어주신 것 중에 있었을 것이다.

Rosemary Bailey, Friedemann Brauer, Jan Busa, Markus Brühwiler,
David Carlisle, José Carlos Santos, Mike Chapman,
Christopher Chin, Carl Cerecke, Chris McCormack, Wim van Dam,
Jan Dittberner, Michael John Downes, David Dureisseix, Elliot,
David Frey, Robin Fairbairns, Jörg| Fischer, Erik Frisk, Frank,
Kasper B. Graversen, Alexandre Guimond, Cyril Goutte,
Greg Gamble, Neil Hammond, Rasmus Borup Hansen,
Joseph Hilferty, Björn Hvittfeldt, Martien Hulsen, Werner Icking,
Jakob, Eric Jacoboni, Alan Jeffrey, Byron Jones, David Jones,
Johannes-Maria Kaltenbach, Michael Koundouros, Andrzej Kawalec,
Alain Kessi, Christian Kern, Jörg Knappen, Kjetil Kjernsmo,
Maik Lehradt, Alexander Mai, Martin Maechler,
Aleksandar S Milosevic, Claus Malten, Kevin Van Maren,
Lenimar Nunes de Andrade, Hubert Partl, John Refling,
Mike Ressler, Brian Ripley, Young U. Ryu, Bernd Rosenlecher,
Chris Rowley, Hanspeter Schmid, Craig Schlenter,
Christopher Sawtell, Geoffrey Swindale, Josef Tkadlec, Didier Verna,
Fabian Wernli, Carl-Gustav Werner, David Woodhouse, Chris York,
Fritz Zaucker, Rick Zaccone, and Mikhail Zotov.

서 문

LaTeX [1]은 과학 및 수학 문서를 작성하는 데 적당한 조판 시스템으로서 대단히 높은 조판 품질을 얻을 수 있게 한다. 또한 단순한 편지글에서 완전한 단행본에 이르기까지 여러 종류의 문서를 만드는 데도 적합하다. LaTeX은 TeX [2]을 조판 엔진으로 사용한다.

이 길지 않은 입문서는 LaTeX 2_ε에 대해서 설명한다. 그리고 LaTeX을 응용해서 쓰는 방법을 충분히 소개할 생각이다. LaTeX 시스템에 대한 완전한 설명을 보려면 [1, 3]을 참조하면 된다.

LaTeX은 PC, Mac, UNIX, VMS 시스템 등 대부분의 컴퓨터에서 동작한다. 대학의 컴퓨터실에는 대개 LaTeX이 이미 설치되어 있어서 즉시 사용할 수 있는 경우가 많을 것이다. 현재 자신이 이용할 수 있는 시스템에 LaTeX이 어떻게 설치되어 있고 어떻게 사용해야 하는지에 대한 안내는 *Local Guide* [4]에 나와 있을 것이다. 만약 LaTeX를 사용하는 데 문제가 있다면 이 책을 권한 사람에게 물어 보라. 이 책은 시스템에 LaTeX를 설치하거나 설정하는 방법에 대해서가 아니라, LaTeX으로 문서 작성하는 방법을 알려주고자 하는 것이다.

이 책은 다섯 개의 장으로 이루어져 있다.

제 1 장 LaTeX 2_ε 문서의 기본 구조를 설명한다. 그리고 LaTeX의 역사에 관해서도 간략히 알게 될 것이다. 이 장을 읽고 나면 LaTeX에 대해서 개략의 지식을 얻게 될 것인데, 이것은 골격에 불과하지만 이어지는 장에서 설명되는 내용과 연결하면 LaTeX 전체에 대해 잘 알 수 있게 될 것이다.

제 2 장 문서를 조판하는 세부사항에 대한 내용이 시작된다. 필수적인 LaTeX 명령과 환경 거의 대부분을 설명한다. 이 장을 읽은 후에는 처음으로 간단한 문서를 작성할 수 있게 될 것이다.

제 3 장 LaTeX으로 수식을 표현하는 방법을 설명한다. 또, 많은 예제를 통해 LaTeX의 가장 주요한 장점인 수식 표현을 어떻게 할 것인지 쉽게 이해하도록 해줄 것이다. 이 장의 끝에는 LaTeX에서 사용할 수 있는 수학 기호들의 목록을 표로 만들어 두었다.

제 4 장 찾아보기와 참고문헌 만들기, EPS 그림 파일 포함하기, 그밖의 다른 유용한 기능 확장에 대하여 설명한다.

제 5 장 LaTeX이 만들어내는 표준 문서 레이아웃을 변경하는, 약간 위험스러울지도 모르는 내용이 들어 있다. 즉, LaTeX이 만들어 내는 아름다운 결과물을 어떻게 하면 엉망으로 만들 수 있는지 알려준다.

순서대로 각 장을 읽도록 하는 것이 중요하다. 이 책은 그다지 크지 않은 규모이다. 특히 예제를 주의깊게 읽어야 한다. 왜냐하면 이 책 전체에 걸쳐 나타나는 예제에 아주 중요한 정보들이 많이 담겨 있기 때문이다.

LaTeX 관련 문서나 자료가 필요하다면 Comprehensive TeX Archive Network(CTAN) 사이트 가운데 하나를 살펴보면 된다. 홈사이트는 http://www.ctan.org이다. 모든 LaTeX 관련 패키지들은 ftp://www.ctan.org ftp 아카이브나 세계 여러 곳에 있는 미러 사이트에서 얻어올 수 있다. 예를 들면, ftp://ctan.tug.org (미국), ftp://ftp.dante.de (독일), ftp://ftp.tex.ac.uk (영국) 가운데 한 곳을 찾아보라. 이 곳들이 아니라도 자신에게 가장 가까운 곳에서 운영하는 미러 사이트를 이용할 수 있다.

이 책의 여러 곳에서 CTAN을 언급하는 것을 볼 수 있을 것이다. 특히 소프트웨어나 문서를 다운로드받아야 하는 것을 지정하는 곳에서는 주로 CTAN을 언급하였는데, 여기서 전체 url을 다 기록하지 않고 다만 `CTAN:`이라고만 쓰고 그 뒤에 CTAN 트리 아래의 적절한 위치를 기록하였다.

만약 자신의 개인 컴퓨터에서 LaTeX를 운영하고 싶다면 `CTAN:/tex-archive/systems`에서 적당한 것을 찾아보면 될 것이다.

만약 이 책의 내용에 추가하거나, 삭제하거나, 변경해야 할 내용이 있다는 생각이 들면 나(저자)에게 알려 달라. 특히 LaTeX 초보자가, 이 개설서 중 어떤 부분이 이해하기 쉽다거나, 더 잘 설명할 수 있는 방법이 있을 것 같다는 등의 제안을 해주는 데 관심이 있다.

Tobias Oetiker `<oetiker@ee.ethz.ch>`

Department of Electrical Engineering,
Swiss Federal Institute of Technology

이 책의 최신 버전은 `CTAN:/tex-archive/info/lshort`에서 찾을 수 있다.

차 례

그림 차례

표 차례

제1장

알아두어야 할 기본사항

이 첫 장의 앞부분은 LaTeX 2ε의 철학과 역사에 대하여 짤막하게 소개하고, 뒷부분에서 LaTeX 문서의 기본 구조를 알아보겠다. 이 장을 읽고나면 LaTeX이 어떻게 작동하는지에 대하여 대강 알게 될 것이다. 이 개략적 지식을 지침으로 하여 계속 읽어가면서 이어지는 각 장에서 소개되는 내용을 보충하면, 전체적인 이해가 보다 쉬워질 것이다.

1.1 어떻게 불러야 하는가?

1.1.1 TeX

TeX은 Donald E. Knuth [2]가 만든 컴퓨터 프로그램으로, 텍스트와 수학식을 조판하기 위해 만들어진 것이다. Knuth가 조판 엔진을 작성하기 시작한 것은 1977년으로, 그 당시 출판업계에 한참 도입되기 시작한 전자 출판 장비의 가능성을 시험하려는 의도도 있었고, 특히 자신의 저서와 논문에까지 영향을 미치게 된 출판물의 전반적 품질 저하 경향을 반전시키고자 하는 희망을 가지고 시작한 일이었다. 오늘날 우리가 사용하는 TeX은 1982년에 발표된 후 몇 년간 조금씩 개선되어 갔다. 1989년에 8비트 문자 지원과 다국어 지원이 추가되었다. 현재의 TeX은 상당히 안정적인 프로그램으로, 다양한 종류의 컴퓨터에서 작동하며, 버그가 없는(virtually bug-free) 프로그램으로 알려져 있다. TeX의 판 번호는 원주율(π)을 따라서 늘어가는데, 현재 3.14159까지 전개되었다.

TeX은 "테흐"(독일어 "Ach"나 스코틀랜드어 "Loch"의 "ch"처럼)로 발음한다.[1] 텍스트(ASCII)로 써야 할 때는 TeX을 `TeX`이라고 쓴다.

[1] 우리나라에서는 "텍"이라고 읽는 것이 관례이다. 이 번역본에서 그렇게 썼다.[역자]

1.1.2 LaTeX

LaTeX은 TeX의 매크로 패키지이다. 이것은 글 쓰는 이로 하여금 미리 정의된 전문적인 레이아웃을 이용하여 고품위의 타이포그래피(조판조정)로 자신의 저작을 조판하고 인쇄하도록 해 준다. LaTeX은 원래 Leslie Lamport [1]에 의하여 작성되었다. TeX 포맷기를 조판 엔진(typesetting engine)으로 사용한다.

1994년에 LaTeX 패키지는 Frank Mittelbach가 주도하는 LaTeX3 팀에 의하여 개선되었다. 여기서 그 동안의 오랜 개선 요구가 받아들여졌고 그 몇 해 전 LaTeX 2.09가 릴리즈된 이래 산발적으로 쌓여 온 패치 버전을 전부 재정비하였다. 새로운 버전은 이전의 것과 구별하기 위하여 LaTeX 2_ε라고 부른다. 이 문서는 LaTeX 2_ε에 대한 것이다.

LaTeX은 "레이텍(Lay-Tech)" 또는 "라텍(Lah-Tech)"으로 읽는다. 텍스트 환경에서는 `LaTeX`이라고 쓴다. LaTeX 2_ε는 "레이텍 투 이(Lay-Tech Two e)"라고 읽고 `LaTeX2e`라고 쓴다.

그림 1.1은 TeX과 LaTeX 2_ε가 어떻게 함께 동작하는지를 보여준다. 이 그림은 Kees van der Laan의 `wots.tex`에서 가져온 것이다.

1.2 기본사항

1.2.1 저자, 디자이너, 조판사

무언가를 출판하려면, 저자는 타자기로 원고를 작성하여 출판사에 넘겨준다. 그 회사의 북디자이너 가운데 한 명이 이 문서의 레이아웃(외양: 문단폭, 글꼴, 제목줄 앞뒤의 여백 …)을 결정한다. 디자이너는 자신의 지시사항을 원고에 써넣은 다음 조판사에게 넘긴다. 조판사는 이 지시사항에 따라 책을 조판한다.

디자이너는 저자가 원고를 쓰면서 염두에 둔 것이 무엇인가를 감안하면서, 그 원고의 내용과 자신의 전문지식에 입각하여 장의 제목줄, 인용문, 예문, 수식 등등을 결정하는 것이다.

LaTeX을 사용하는 경우, LaTeX이 디자이너 역할을 한다. 그리고 TeX을 조판기로 이용한다. 그러나 LaTeX은 컴퓨터 프로그램에 지나지 않으므로, (사람인 디자이너에게 맡기는 경우보다) 더 많은 지시사항을 알려주어야 할 것이다. 저자는 자기 글의 논리적 구조를 나타내는 정보를 텍스트와 함께 추가로 제공하여야 하는데, 이 정보는 "LaTeX 명령"이라는 형태로 텍스트에 함께 써넣는다.

이러한 방식은 이른바 WYSIWYG[2]라고 알려진, *MS Word*나 아래아한글[3] 같은 요즘 거의 모든 워드 워드 프로세서가 취하는 방법과는 상당히 다른 것이

[2] What you see is what you get. 화면에 보이는 대로 출력된다는 말.
[3] 원문에는 *Corel Word-Perfect*[역주]

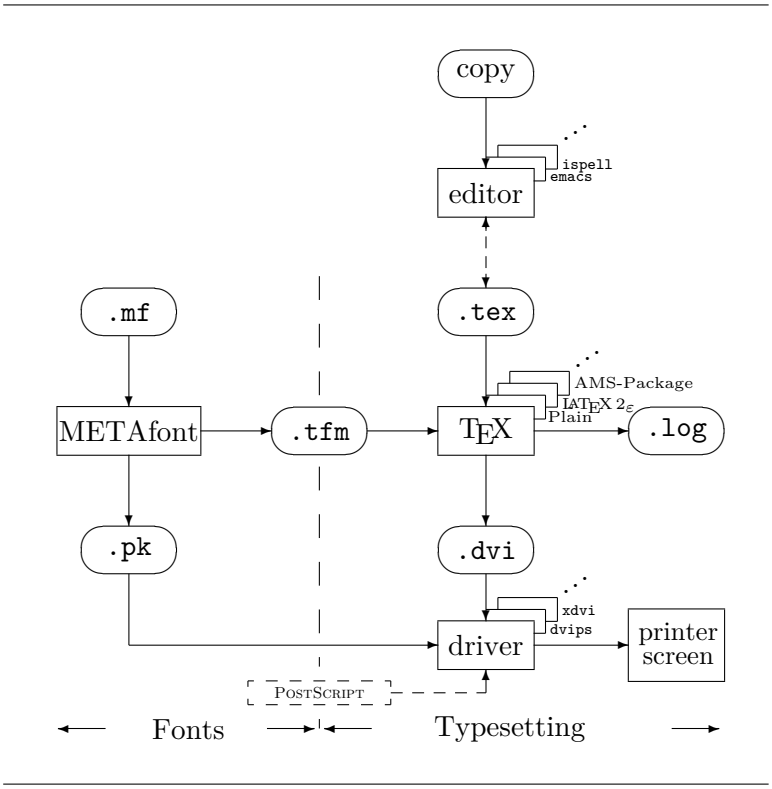


그림 1.1: TEX System의 구성

다. WYSIWYG 방식은, 저자가 컴퓨터에 텍스트를 타이핑하면서 대화식으로 문서의 레이아웃을 조정해간다. 즉, 자신의 최종 작업 결과가 출력되었을 때의 모양을 화면으로 보면서 작업할 수 있는 것이다.

LaTeX을 사용할 때는 텍스트를 입력하는 동안 최종 출력물의 모양을 보지는 못한다. 그러나 그 입력 파일을 LaTeX으로 처리(컴파일)한 후에는 최종 출력 결과를 화면으로 미리보기할 수 있다. 미리보기를 통하여 어떤 부분을 수정해야 할지 알 수 있으므로 필요한 수정을 가한 다음, 마지막으로 프린터로 보낸다.

1.2.2 레이아웃 디자인

타이포그래피 디자인은 전문적인 일이다. 이 분야에 익숙지 않은 저자들이 자주 저지르는 심각한 실수는, 책을 디자인하는 것이 대부분 미관의 문제라고 생각하는 것이다. "디자인이 잘 된 거? 문서가 보기 좋으면 되는 거 아냐?" 그러나, 문서라는 것은 읽히기 위해 있는 것이지 화랑에 걸어두려고 있는 것이 아니다. 가독성과 이해가능성은 그 외관의 아름다움보다 훨씬 더 중요하다. 예를 들어보자.

- 글꼴 크기와 제목 번호붙이기는 독자에게 장/절의 구조가 명확하게 전달되도록 선택되어야 한다.
- 행 길이는 독자의 시야를 넘어서지 않을 정도로 짧아야 하지만, 쪽를 보기좋게 채울 정도로는 길어야 한다.

WYSIWYG 시스템을 사용하는 저자들은 외관상 보기에는 좋지만 도대체 구조적이지 못하고 일관성도 없는 문서를 만들어내는 경우가 많다. LaTeX은 저자 자신의 문서에 논리적인 구조를 선언하게 함으로써 이러한 구조 오류를 피하게 해준다. 그런 다음 이 구조에 가장 적당한 레이아웃을 LaTeX이 선택하는 것이다.

1.2.3 장점과 단점

WYSIWYG에 익숙한 사람들이 LaTeX을 사용하는 사람을 만나거나 하면 "일반 워드 프로세서에 비해 LaTeX이 가진 장점이 무엇이냐"거나, 반대로 워드 프로세서의 좋은 점이 무엇이냐에 대해 논쟁이 벌어지는 일이 종종 있다. 이런 논쟁이 벌어지면 자리를 피하는 것이 상책이다. 왜냐하면 이런 식의 논의는 걷잡을 수 없게 비화하기 일쑤이기 때문인데, 그래도 가끔은 피할 수 없을 때가 있는 법…

그래서 여기 탄약 몇 발을 마련해 둔다. LaTeX이 일반 워드 프로세서에 비해서 더 좋은 점은 다음과 같다:

- 전문적으로 디자인된 레이아웃을 사용할 수 있다. 그래서 문서가 "인쇄된" 것과 거의 동일하다.

- 수학식의 조판이 매우 쉽다.
- 사용자는 문서의 논리적 구조를 지시하는 몇 가지 기억하기 쉬운 명령어들만 익히면 된다. 이 명령들은 대부분 눈으로 보이는 문서의 모양을 주물러야 하는 것이 아니다.[4]
- 각주, 상호참조, 목차, 참고문헌 등 매우 복잡한 구조들도 아주 쉽게 만들어진다.
- LaTeX기본 패키지 이외에도 여러가지 조판상의 필요를 충족시키는 무료의 추가 패키지들이 많다. PostScript 그림을 포함하게 하거나 표준을 정확하게 지키는 참고문헌을 작성하게 하거나 하는 패키지들을 예로 들 수 있겠다. 이 추가 패키지들은 대부분 *The LaTeX Companion* [3]에 설명되어 있다.
- LaTeX은 구조화가 잘 된 문서를 작성하도록 저자를 유도한다. LaTeX의 동작 방식이 바로 그것이다—구조를 지정하는 것.
- LaTeX 2_ε의 조판 엔진인 TeX은 매우 이식성이 높고, 무료이다. 그러므로 TeX 시스템은 거의 모든 하드웨어 플랫폼에서 실행가능하다.

LaTeX은 단점도 가지고 있다. 내 입장에서 그 중 사리에 맞아 보이는 것은 별로 없지만, 그런 단점을 수백 가지 지적해 내는 사람도 틀림없이 있는 것이다. ;-)

- LaTeX은 아무 생각 없는 머리가 빈 사람들하고는 잘 맞지 않는다…
- 미리 만들어진 레이아웃 인자의 기정값을 조절해가면서 쓸 수 있기는 하지만, 완전히 새로운 레이아웃을 설계하는 것은 어렵고 시간이 많이 드는 일이다.[5]
- 구조화되지 않고 비체계적인 문서를 작성하기가 너무나 어렵다.
- 비록 첫 단계를 지났다하더라도, 논리적인 마크업(Logical Markup)의 개념을 완전히 이해하는 것은 불가능할 수도 있다.

1.3 LaTeX 입력 파일

LaTeX에 의하여 처리되는 것은 텍스트 파일(plain `ASCII` text)이다.[6] 어떤 텍스트 편집기(에디터)로도 만들 수 있다. 여기에는 문서의 내용은 물론이고, LaTeX에게 텍스트를 조판하는 방법에 관하여 알려주는 지시사항들도 기록된다.

[4] 즉, 논리적 구조를 선언하는 것이지 시각적 모양을 조절하는 것이 아니라는 뜻.[역주]

[5] 소문에 의하면 앞으로 나올 LaTeX3 시스템에서는 이것이 주안점이 될 거라 한다.

[6] 우리가 말하는 "텍스트 파일"은 엄밀히 말해서 ASCII가 아니다. 여기서는 plain text 파일이라는 의미로 쓰겠다. [역자]

1.3.1 공백

"공백문자(whitespace characters)", 즉 빈 칸(blank), 탭(tab) 등은 LaTeX에서 모두 동일하게 "스페이스"로 처리된다. 계속되는 여러 개의 공백문자들은 하나의 "스페이스"로 취급된다. 행의 첫머리에 있는 공백문자들은 무시되고 한 번의 줄바꿈(개행: linebreak) 역시 "공백문자"로 간주된다.

두 줄 사이에 빈 줄을 하나 넣으면 문단(paragraph)의 끝을 나타낸다. 여러 개의 빈 줄은 하나의 빈 줄과 같다. 아래 예를 보자. 왼쪽은 입력 파일에 적힌 텍스트의 내용이고 오른쪽은 그 출력된 형태를 나타낸다.

```
It does not matter whether you
enter one or several     spaces
after a word.

An empty line starts a new
paragraph.
```

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.

1.3.2 특수문자

다음 기호들은 유보된 문자들로서 LaTeX에서 특별한 의미를 갖거나 어떤 글꼴로도 나타낼 수 없는 것들이다. 이것을 텍스트에 직접 써넣는다면 대개 프린트되지 않을 것이며 그럼에도 LaTeX에게 강제로 실행하도록 하면 원치 않는 결과를 초래할 것이다.

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

앞으로 보게 되겠지만, 이러한 문자를 문서에서 사용하려면 백슬래시(\)를 더해 주어야 한다.

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

$ % ^ & _ { } ~

그밖의 많은 기호와 부호, 수학식에서 쓰이는 문자, 억양 표시가 붙은 문자들은 특별한 명령을 써서 출력한다. 백슬래시 문자 \는 백슬래시를 그 앞에 하나 더 붙인다고(\\) 얻어지는 것이 아니다. 이 부호는 줄바꿈(개행)에 사용된다.[7]

1.3.3 LaTeX 명령

LaTeX 명령은 대소문자를 구별하며 다음과 같은 두 가지 형태 가운데 하나를 취한다:

[7] '\'를 얻으려면 `$\backslash$` 명령을 사용한다.

- 백슬래시 \로 시작하여 문자만으로 이루어진 이름을 갖는다. 명령 이름은 공백이나 숫자 또는 '문자 아닌 것'이 오면 끝난다.
- 백슬래시 다음에 딱 한 개의 특별한 글자가 온다.

LaTeX은 명령 다음의 공백문자를 무시한다. 명령 다음에 빈 칸을 두고 싶으면 {}와 빈 칸을 써넣거나 명령 이름 뒤에 특별한 칸 띄우기 명령을 써야 한다. {}는 LaTeX이 명령 이름 다음에 나오는 빈 칸들을 "먹어버리지"(무시하지) 않도록 만든다.

```
I read that Knuth divides the
people working with \TeX{} into
\TeX{}nicians and \TeX perts.\\
Today is \today.
```

I read that Knuth divides the people working with TeX into TeXnicians and TeXperts.
Today is 2019년 5월 13일.

인자(parameter)가 필요한 명령도 있다. 인자는 명령 이름 다음의 중괄호 { } 속에 써넣어야 한다. 어떤 명령에는 선택 인자(optional parameters)가 필요한 경우도 있는데, 명령 바로 다음에 각진괄호(square bracket) [ ]를 쓰고 그 안에 써넣는 것이 이것이다. 다음은 LaTeX 명령을 사용하는 보기이다. 무슨 말인지 모르겠더라도 걱정할 것 없다. 나중에 설명이 될 것이다.

```
You can \textsl{lean} on me!
```

You can *lean* on me!

```
Please, start a new line
right here!\newline
Thank you!
```

Please, start a new line right here!
Thank you!

1.3.4 주석

LaTeX은 입력 파일을 처리해가다가 % 문자를 만나면 그 줄(행)의 나머지 부분과 줄바꿈을 무시한다. 그리고 다음 줄 첫머리에 오는 공백문자도 모두 무시한다.

이것은 입력 파일 작성시 최종 인쇄판에는 나타나지 않는 주석문을 쓰고 싶을 때 사용할 수 있다.

```
This is an % stupid
% Better: instructive <----
example: Supercal%
              ifragilist%
    icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

% 문자는 공백문자나 줄바꿈 문자가 허용되지 않는 긴 줄을 나누어 입력할 수 있게 해준다.

좀 더 긴 주석을 쓰려면 verbatim 패키지에서 제공하는 comment환경을 사용해야 한다. 이 말은, 문서의 전처리부(preamble)에 \usepackage{verbatim}라는 명령을 추가한 다음 comment 환경 안에 글을 쓰라는 것이다.

```
This is another
\begin{comment}
rather stupid,
but helpful
\end{comment}
example for embedding
comments in your document.
```

This is another example for embedding comments in your document.

주의할 점은 이 방법이 예컨대 수학식 같은 복잡한 환경 안에서는 제대로 동작하지 않는다는 것이다.

1.4 입력 파일의 구조

LaTeX이 입력 파일을 처리할 때는 그 파일이 일정한 구조를 따르고 있다고 가정한다. 그러므로 모든 입력 파일은 다음 명령으로 시작해야 한다.

```
\documentclass{...}
```

이 명령은 지금 쓰려 하는 문서가 어떤 종류의 것인지 설정하는 것이다. 이 다음에 전체 문서의 모양(스타일)에 영향을 주는 명령들을 포함하거나 LaTeX 시스템에 새로운 기능을 추가하는 패키지들을 포함할 수도 있다. 패키지를 포함할 때는 다음과 같은 형태의 명령을 쓴다.

```
\usepackage{...}
```

이 설정 과정이 모두 끝난 뒤에,[8] 문서의 주요부가 시작된다. 다음이 문서 주요부의 시작을 나타내는 명령이다.

```
\begin{document}
```

이제 문장을 몇 가지 LaTeX 명령과 섞어가면서 입력하면 된다. 문서의 끝에는

```
\end{document}
```

[8] \documentclass와 \begin{document} 사이에 오는 부분을 전처리 부분(*preamble*)이라 한다.

```
\documentclass{article}
\begin{document}
Small is beautiful.
\end{document}
```

그림 1.2: 최소한의 LaTeX 파일

```
\documentclass[a4paper,11pt]{article}
% define the title
\author{H.~Partl}
\title{Minimalism}
\begin{document}
% generates the title
\maketitle
% insert the table of contents
\tableofcontents
\section{Start}
Well, and here begins my lovely article.
\section{End}
\ldots{} and here it ends.
\end{document}
```

그림 1.3: 실제 학술지 논문의 보기

명령을 추가한다. 이것은 LaTeX에게 여기가 처리해야 하는 부분의 끝임을 알려준다. 이 명령 이후에 나오는 것은 모두 무시된다.

그림 1.2에 보인 것은 최소한의 LaTeX 파일 내용이다. 이보다 좀 복잡한 입력 파일이 그림 1.3에 있다.

1.5 명령행 작업의 전형적인 방식

9 페이지에 예시된 정말 소략한 LaTeX 입력 파일을 가지고 뭘 어떻게 하면 좋은지 황당해하고 있는가? 몇 가지 도움을 주고자 한다. LaTeX 자체는 GUI 나 예쁘장한 누름버튼 같은 것이 전혀 없는 프로그램이다. 그것은 그냥 입력 파일을 처리하는 프로그램일 뿐이다. 어떤 LaTeX 시스템은 입력 파일을 컴파일할 수 있는 클릭 버튼을 가진 GUI 방식의 프런트 엔드(front end) 프로그램을 설치해주기도 한다. 그러나, 진짜배기 사용자는 "클릭"하지 않는다. 여기서는 텍스트 기반 시스템에서 LaTeX을 어떻게 다루어야 입력 파일을 컴파일할 수

있는지 보여주려 한다. LaTeX 시스템이 컴퓨터에 이미 설치되어 있는 것으로 가정하고 설명한다.

1. LaTeX 입력 파일을 만들고 편집한다. 반드시 plain 아스키 텍스트 파일이어야 한다. Unix 시스템에서는 어떤 에디터를 사용하더라도 아스키 텍스트 파일을 만들어준다. 윈도 시스템에서는 파일을 저장할 때 ASCII 또는 *plain text* 포맷으로 저장하여야 한다. 파일 이름을 선택할 때 확장명을 `.tex`으로 주는 것을 잊지 말도록 하라.

2. 그 입력파일에 대하여 LaTeX을 실행한다. 성공적으로 실행되고 나면 `.dvi` 파일을 얻을 수 있다.

   ```
   latex foo.tex
   ```

3. 이제 DVI 파일을 화면보기 한다.

   ```
   xdvi foo.dvi
   ```

   또는 PS 파일로 변환한다.

   ```
   dvips -Pcmz foo.dvi -o foo.ps
   ```

   `xdvi`와 `dvips`는 `.dvi` 파일을 처리할 수 있는 오픈소스 툴들이다. `xdvi`는 X11 환경에서 `.dvi` 파일을 화면에 그려주는 것이고, `dvips`는 프린트할 수 있도록 PostScript 파일을 만들어준다. Unix 시스템이 아니라면 플랫폼에 적합한 `.dvi` 파일 처리기가 별도로 제공될 것이다.[9]

1.6 문서의 레이아웃

1.6.1 문서 클래스

LaTeX이 입력되는 파일을 처리할 때 가장 먼저 제공되어야 하는 정보는 저자가 만들려는 문서의 유형(type)이다. 이것은 `\documentclass` 명령으로 지정한다.

`\documentclass[`*options*`]{`*class*`}`

여기서 *class*라는 인자는 만들어질 문서의 유형을 지시한다. 표 1.1에 이 책에서 설명하는 문서 클래스들을 요약해 놓았다. LaTeX 2_ε 배포본에는 편지나 슬라이드 등의 문서를 위한 클래스가 더 포함되어 있다. *option* 인자는 문서 클래스의 동작을 사용자의 의도에 맞게 조절하는 데 쓰인다. 여러 개의 옵션은 쉼표로 구분한다. 표준적 문서 클래스에 사용되는 일반적 옵션을 표 1.2에 요약하였다.

보기 : LaTeX 문서의 첫 줄이 다음과 같이 시작되었다면,

```
\documentclass[11pt,twoside,a4paper]{article}
```

이것은 LaTeX에게 문서를 *article*로 작성하고 기본 글꼴 크기를 11포인트로 하고, A4 용지에 양면 인쇄하기에 적당하도록 레이아웃을 잡으라는 것을 지시하고 있다.

1.6.2 패키지

글을 써가다 보면, 기본 LaTeX만으로는 해결할 수 없는 문제를 만날 때가 있을 것이다. 그림을 포함하려 하거나, 채색 글씨를 쓰고 싶을 때, 또는 문서에 소스 코드를 삽입해야 할 경우, LaTeX의 기능을 향상시켜야 할 필요를 느끼게 된다. 이러한 기능 향상은 패키지를 통해서 이루어진다. 다음 명령을 써서 패키지를 사용 가능하게 한다.

`\usepackage[`*options*`]{`*package*`}`

여기서 *package*란, 패키지의 이름을 가리키고, *options*란 패키지가 수행해야 할 특정 기능을 나타내는 지시어들을 말한다. LaTeX 2_ε 기본 배포판에 함께 따라오는 패키지들도 있다. (표 1.3을 참조.) 그밖에 독자적으로 제공되는 패키지도 있다. 자신의 컴퓨터에 어떤 패키지들이 설치되어 있는지 알고 싶으면 거기에 있는 *Local Guide* [4]를 찾아보면 된다. LaTeX 패키지에 대해서는 *The LaTeX*

[9] `dvips`는 윈도에서도 쓸 수 있다. `xdvi` 대신 윈도에서는 TeX 설치판에 따라 `windvi` 또는 `yap` 등이 제공된다.—[역자]

표 1.1: 문서 클래스들

`article`	과학 학술지, 프레젠테이션, 짧은 보고서, 프로그램 문서, 초대장 등에 쓰이는 아티클용 클래스
`report`	여러 장으로 이루어진 긴 보고서, 작은 책, 박사학위 논문 등에 쓰이는 클래스
`book`	진짜 책을 만들기 위한 클래스.
`slides`	슬라이드 제작용 클래스. 이 클래스는 큰 산세리프 글꼴을 사용한다. 이것 대신 FoilTeX[a]의 사용을 고려해 볼 만도 함.

[a] `CTAN:/tex-archive/macros/latex/contrib/supported/foiltex`

표 1.2: 문서 클래스의 옵션

`10pt`, `11pt`, `12pt` 문서의 기본 글꼴 크기를 설정한다. 아무 옵션도 주지 않으면 `10pt`로 간주된다.

`a4paper`, `letterpaper`, ... 종이 크기를 지정한다. 기본 크기는 `letterpaper`이다. 이외에도, `a5paper`, `b5paper`, `executivepaper`, `legalpaper` 등을 지정할 수 있다..

`fleqn` 수식을 가운데 정렬하지 않고 왼쪽 정렬하도록 정한다.

`leqno` 수식의 번호를 오른쪽이 아니라 왼쪽에 붙이도록 한다.

`titlepage`, `notitlepage` 표지를 만든 다음 새로운 쪽으로 시작할 것인지 그렇지 않을 것인지를 지정한다. `article` 클래스는 새로운 쪽으로 시작하지 않는 것이 기본값이다. `report`와 `book`은 표지를 한 쪽으로 만드는 것이 기본값이다.

`twocolumn` LaTeX에게 2단으로 조판하라고 지시하는 것.

`twoside, oneside` 양면인쇄용 또는 단면인쇄용 출력물을 만들라고 지시하는 것이다. `article`과 `report`는 단면, `book` 클래스는 양면이 기본값이다. 이 옵션은 문서의 모양에 관한 것일 뿐이다. 즉, `twoside` 옵션을 주었다고 해서 프린터가 실제로 양면으로 출력해주는 것은 아니다.

`openright, openany` 새로운 장을 홀수쪽에서만 시작할 것인지 홀/짝 구분 없이 바로 다음 쪽에서 시작할 것인지를 지정하는 것이다. 이 옵션은 장(chapter)이라는 개념이 없는 `article` 클래스에서는 동작하지 않는다. `report` 클래스는 다음 쪽에서 시작하는 것이 기본값으로 되어 있고 `book` 클래스에서는 홀수쪽에서 시작하는 것이 기본값이다.

Companion [3]을 보면 많은 것을 알 수 있을 것이다. 이 책은 여러 패키지에 대한 설명은 물론이고 LaTeX 2$_\varepsilon$를 확장하기 위해 자신이 직접 패키지를 작성하는 방법에 대해서도 설명하고 있다.

표 1.3: LaTeX과 함께 배포되는 패키지 몇 가지

doc LaTeX 프로그램의 문서화를 가능하게 한다.
`doc.dtx`[a]와 *The LaTeX Companion* [3]에 설명.

exscale 확장 수학 글꼴을 크기별로 조절할 수 있게 하는 기능을 제공한다.
`ltexscale.dtx`에 설명.

fontenc LaTeX이 어떤 글꼴 인코딩을 사용할 것인지 지정한다.
`ltoutenc.dtx`에 설명.

ifthen 다음과 같은 문장 형식을 지원한다.
'if…then do…otherwise do….'
`ifthen.dtx`와 *The LaTeX Companion* [3]에 설명.

latexsym LaTeX 기호문자(심볼) 글꼴 사용을 가능하게 하려면 `latexsym` 패키지를 사용해야 한다. `latexsym.dtx`와 *The LaTeX Companion* [3]에 설명.

makeidx 찾아보기(index)를 만드는 명령을 제공한다. 이 글 4.3 절과 *The LaTeX Companion* [3]에 설명.

syntonly 문서를 처리(컴파일)하지만 조판하지는 않도록 한다.

inputenc 입력 파일의 인코딩 방법을 지정할 수 있게 해 준다. 예를 들면, ASCII, ISO Latin-1, ISO Latin-2, 437/850 IBM 코드 페이지, 애플 매킨토시, 넥스트, ANSI-윈도, 및 사용자 정의 인코딩 등. `inputenc.dtx`에 설명.

[a] 이 파일은 시스템에 틀림없이 설치되어 있을 것이다. `*.dtx` 파일로부터 `dvi` 파일을 얻으려면 쓰기 권한이 있는 디렉토리에서 명령행에 `latex doc.dtx`라고 쓰면 된다. 이 표에서 언급된 다른 파일도 마찬가지다.

1.6.3 쪽 양식

LaTeX에는 세 가지 쪽머리글/쪽바닥글 조합(이것을 쪽 양식(page style)이라 한다.)이 미리 정의되어 있다.

`\pagestyle{`*style*`}`

이 명령의 *style* 인자가 이 가운데 어느 것을 사용할지 지정한다. 표 1.4에 미리 정의된 쪽 양식을 요약해 두었다.

표 1.4: LaTeX에서 미리 정의된 쪽 양식

`plain`	쪽바닥글 중앙에 쪽 번호를 붙인다. 이것이 기본값이다.
`headings`	현재의 장 제목과 쪽 번호를 각 쪽머리글에 인쇄하고 쪽바닥글은 비워둔다. (이 문서에서 사용한 스타일이다.)
`empty`	쪽머리글/쪽바닥글을 모두 비운다.

현재 쪽의 쪽 양식을 바꾸는 데는 다음 명령을 쓴다.

`\thispagestyle{`*style*`}`

자신만의 쪽머리글과 쪽바닥글을 만드는 방법이 *The LaTeX Companion* [3]에 나와 있고, 이 책 70 쪽 4.4 절에서도 찾아볼 수 있다.

1.7 작업 중에 만나게 되는 파일들

LaTeX으로 작업하다 보면, 곧 다양한 확장자를 가진 파일들의 미궁 속에 빠져든 자신을 보게 될 것이다. 뭐가 뭔지 알 수도 없다. 아래에 TeX으로 작업할 때 만날 수 있는 여러 가지 파일 형식에 대한 리스트를 제시하였다. 모든 파일 형식을 모두 망라한 것은 아니라는 점에 주의하라. 만약 중요한 것인데 빠진 것이 있다고 생각되면 알려주기 바란다.

.tex LaTeX 혹은 TeX의 입력 파일이다. `latex`으로 컴파일한다.

.sty LaTeX 매크로 패키지 파일이다. `\usepackage` 명령을 사용해서 LaTeX 문서에 적재할 수 있다.

.dtx 문서화된 TeX(Documented TeX) 파일. LaTeX 스타일 파일의 주 배포 형식이다. .dtx 파일을 컴파일하면, .dtx 파일에 포함된 LaTeX 패키지의 매크로 코드를 문서화된 형태로 볼 수 있다.

.ins 같은 이름의 .dtx 파일에 포함된 매크로 코드를 풀어서 설치해주는 역할을 한다. LaTeX 패키지를 네트웍을 통해서 다운받았다면, 대부분 .dtx 파일과 .ins 파일로 이루어져 있을 것이다. .ins 파일에 대하여 LaTeX을 실행하면 .dtx 파일을 풀어서 매크로 스타일 파일을 얻을 수 있다.

.cls 문서의 양식을 정의하는 클래스 파일이다. `\documentclass` 명령의 인자로 쓰여서 그 문서의 외양을 결정하게 된다.

다음 파일들은 LaTeX을 실행할 때 볼 수 있는 파일들이다.

.dvi 장치 독립(device independent) 파일이다. LaTeX 컴파일을 실행한 결과 얻어지는 주요 파일이다. DVI 미리보기 프로그램를 이용해서 내용을 화면으로 볼 수 있고 `dvips` 또는 그와 유사한 애플리케이션을 사용하여 프린터로 보낼 수도 있다.

.log 마지막 컴파일을 실행했을 때 무슨 일이 일어났던가에 대한 설명을 보여준다.

.toc 각 장/절의 타이틀을 모아둔 것이다. 컴파일을 한 번 더 실행하면 컴파일러는 여기에 저장된 정보를 읽어서 목차를 만든다.

.lof .toc와 비슷한 것으로 그림 목차를 만드는 데 이용된다.

.lot 마찬가지로, 표 목차를 만들기 위한 것이다.

.aux 한 번 컴파일이 이루어진 후에 다음번에 실행될 때 전달해 줄 정보를 담고 있는 또 하나의 파일이다. 무엇보다도, .aux 파일은 상호참조에 필요한 관련 정보를 담고 있다.

.idx 만약 찾아보기에 대한 설정이 사용되었다면, LaTeX은 찾아보기용으로 쓰일 모든 단어를 이 파일에 저장한다. 이 파일은 `makeindex`로 처리하여야 한다. 찾아보기에 대해서는 68 페이지의 제 4.3 장을 참고하라.

.ind .idx파일에 `makeindex`를 실행하면 만들어지는 파일로서, 다음 번 컴파일 때 문서에 포함될 내용을 담고 있다.

.ilg `makeindex`가 무엇을 어떻게 처리하였는가에 대한 로그파일이다.

1.8 대규모 프로젝트

큰 문서를 작업하려면 입력 파일을 몇 개 부분으로 나누어 작업하고 싶을 때가 있다. LaTeX은 이를 위해 두 가지 명령을 제공한다.

`\include{`*filename*`}`

이 명령을 사용하면 문서의 일부에 *filename.tex*라 불리는 다른 파일의 내용을

삽입할 수 있다. LaTeX이 이 명령을 만나면 *filename*으로 삽입되는 부분을 처리할 때 새로운 쪽으로 시작한다는 점에 주의하라.

또 하나의 명령은 전처리(preamble) 부분에서 쓰일 수 있다. 다음 명령은 `\include` 명령에 지정된 파일이 여기에 열거된 것 중 하나일 때만 그 파일을 삽입하라고 LaTeX에게 지시하는 것이다.

`\includeonly{`*filename*`,`*filename*`,⋯}`

이 명령이 문서의 전처리 부분에서 실행된 후에는 `\include` 명령으로 삽입하는 파일들 가운데 여기서 나열된 것만이 실제로 삽입될 것이다. 파일 이름과 쉼표 사이에 빈 공간이 없어야 한다는 데 주의한다.

`\include` 명령은 파일을 삽입할 때 새로운 쪽을 만들고 거기에서부터 조판하기 시작한다. `\includeonly`를 쓰는 경우 이렇게 하는 편이 나을 수 있다. 왜냐하면 include된 파일 중에 몇 개를 제외하더라도 쪽 나눔 위치가 이동하지 않기 때문이다. 항상 새로운 쪽으로 시작하는 것이 바람직하지 못할 때는 다음의

`\input{`*filename*`}`

명령을 쓰면 된다. 이 명령은 지정된 파일을 포함시키기만 할 뿐이고 페이지 조절이나 문자 추가를 전혀 하지 않으므로 삽입된 위치에서 새로운 쪽으로 나누지 않는다.

문서를 빠르게 처리하게 할려면 syntonly 패키지를 사용하라. 이 패키지는 문서 전체를 읽으며 문법과 명령어가 적절한지를 검사하지만, (DVI) 출력물을 생성하지 않는다. LaTeX은 이 모드에서 상당히 빠르게 동작하므로 귀중한 시간을 아낄 수 있다. 사용법은 아주 간단한다:

```
\usepackage{syntonly}
\syntaxonly
```

실제 출력물을 만들고 싶을 때는, 두번째 줄을 주석처리(퍼센트 표시 `%`를 붙이는 것) 하면 된다.

제2장

텍스트의 조판

앞 장에서 LaTeX 2$_\varepsilon$ 문서를 만들기 위한 기본적인 사항을 알게 되었을 것이다. 이 장에서는 진짜 문서를 만들기 위해서 알아야 할 '구조'의 나머지 사항에 관해 설명한다.

2.1 텍스트와 언어의 구조

글쓰기의 주요 초점(현대의 일부 '무작정 다르게' 문학[1]을 논외로 한다면)은 생각(아이디어)과 정보, 지식을 독자에게 전달하는 것이다. 독자들의 이해가 수월해지려면 이 생각(아이디어)들이 잘 구조화되어 있어야 할 것이다. 그리고 구조를 보다 쉽게 보고 느끼게 하려면 조판 출력(타이포그래피) 형태가 내용의 논리적 · 의미론적 구조를 잘 반영하고 있어야 한다.

LaTeX이 다른 조판 시스템과 다른 점은, 사용자가 텍스트의 논리적 · 의미론적 구조를 지시하여야 한다는 것이다. 그러면 LaTeX은 지정된 문서 클래스 파일과 여러 스타일 파일에 정의된 "규칙"에 따라서 텍스트의 조판 출력(타이포그래피) 형식을 만들어낸다.

LaTeX에서, 그리고 모든 조판 작업(typography)에서 가장 중요한 텍스트 단위는 문단(paragraph)이다. 이것을 "텍스트 단위"라고 말하는 이유는, 문단이 한 가지 일관된 생각이나 개념을 반영하는 조판상의 형식이기 때문이다. 다음 몇 절에서 배우게 될 것이 이와 관련된 사항들이다. 예를 들면, 줄바꿈을 하기 위해 \\를 쓰는 법, 문단을 나누기 위해서 원본 입력 파일에 빈 줄을 넣는 방법 같은 것들이다. 따라서, 새로운 문단은 새로운 생각이 전개되는 곳에서 시작해야 한다. 그렇지 않다면 그냥 줄바꿈만이 쓰여야 한다. 문단을 나누어야 할지 어떨지 확신이 서지 않으면, 글이란 사고와 사상을 전달하는 매체라는 관점에서 자신의 텍스트를 살펴보라. 문단을 나누었는데 이전의 생각이 연속되고

[1] DAAC(Different At All Cost). 스위스식 독일어 UVA(Um's Verrecken Anders)의 영어 표현.

있다면 그 문단 나눔을 없애야 한다. 같은 문단 안에서 완전히 새로운 생각이 전개되는 곳이 있다면 그곳에서 문단을 나누어 주어야 한다.

많은 사람들이 문단 나누기를 제대로 하는 게 얼마나 중요한지를 과소평가하고 있다. 심지어 문단 나누기의 의미가 무엇인지조차 모르는 사람도 있다. 특히 LaTeX을 쓰는 사람 중에는, 자신이 문단을 나누고 있는지도 모르는 채로 문단을 나누는 경우도 많다. 이런 실수는 특히 글 속에 수식이 사용될 경우 저지르기 쉽다. 다음 예를 보면서, 왜 어떤 때는 수식 전후로 빈 줄(문단 나눔)이 사용되고 또 어떤 때는 사용되지 않는지 생각해 보라. (만약 여기서 쓰인 명령들을 아직 충분히 알지 못해서 이 예가 무엇을 하자는 것인지 이해할 수 없다면, 이 장과 다음 장을 읽은 다음에 이 절로 다시 돌아와서 읽어보기 바란다.)

```
% Example 1
\ldots when Einstein introduced his formula
\begin{equation}
  e = m \cdot c^2 \; ,
\end{equation}
which is at the same time the most widely known
and the least well understood physical formula.


% Example 2
\ldots from which follows Kirchoff's current law:
\begin{equation}
  \sum_{k=1}^{n} I_k = 0 \; .
\end{equation}

Kirchhoff's voltage law can be derived \ldots


% Example 3
\ldots which has several advantages.

\begin{equation}
  I_D = I_F - I_R
\end{equation}
is the core of a very different transistor model. \ldots
```

(역자의 보충) 한국어로 글을 쓸 때도 새로운 문단을 시작하는 부분에 수식이 있을 때 주의해야 하는 것은 마찬가지라고 하겠다. 다음 두 가지 예를 번역해 둔다.

예 1

```
\ldots 아인슈타인이 도입한 다음 공식,
\begin{equation}
  e = m \cdot c^2 \;
\end{equation}
은 가장 널리 알려진 것이면서 동시에 가장 이해하는 사람이
적은 공식이다.
```

예 2

```
\ldots로부터 키르히호프의 전류 법칙이 도출된다.
\begin{equation}
  \sum_{k=1}^{n} I_k = 0 \; .
\end{equation}

키르히호프의 전압 법칙은 다음과 같이 유도할 수 있는데 \ldots
```

문단 다음으로 더 작은 글의 단위는 문장(sentence)이다. 영문 텍스트에서는 약어 기호로 쓰인 마침표보다 문장의 끝을 나타내는 마침표의 뒤에 더 큰 공백을 둔다. LaTeX은 저자가 실제로 의도한 것이 문장의 끝이었는지 약어 기호였는지를 스스로 판단한다. 만약 LaTeX의 판단이 잘못된 경우라면, 저자가 직접 LaTeX에게 자신이 원하는 바가 무엇인지를 알려주어야 한다. 이 점에 대해 이 장의 후반부에서 설명될 것이다.

나아가, 텍스트의 구조화는 문장의 일부분에까지 적용된다. 대부분의 언어에서 구두법은 상당히 복잡하다. 그러나, 영어나 독일어는 물론이고 어떤 언어라도, 구두점이 무엇을 나타내는 것인지 생각하기만 한다면 그 구두점을 제 자리에 정확하게 찍는 것은 별로 어려운 일이 아니다. 만약 어디에다가 구두점을 찍어야 하는지 확신이 서지 않으면, 그 문장을 소리내어 읽어보고, 모든 구두점마다 짧게 한 숨 멈추어 보는 것이 좋겠다. 그것이 어색한 곳이 있으면, 그 부분의 구두점을 지우도록 하라. 또 만약 어떤 곳에서 숨을 쉬거나 짧은 기식(stop)을 두는 편이 낫겠다는 느낌이 든다면, 그 곳에 구두점을 찍으면 된다.

끝으로, 텍스트의 문단들도 보다 높은 수준에서 논리적으로 구조화되어야 한다. 즉, 문단을 모아서 장(chapter), 절(section), 소절(subsection) 등으로 배치하여야 하는 것이다. 이러한 높은 수준의 구조화가 어떤 식으로 실현되는가는 자명하다. 왜냐하면 예를 들어, 절의 제목을 나타내기 위해 `\section{The Structure of Text and Language}`라고 입력한다면 그 조판 효과가 어떠하리라는 것은 그 자체로 분명하기 때문이다.

2.2 줄바꿈과 문단나눔

2.2.1 문단 정렬

단행본은 각 행이 똑같은 길이를 가지도록 조판되는 경우가 많다. LaTeX은 문단 내용 전체를 최적화하여 단어 사이에 적절한 줄바꿈과 공백을 삽입한다. 필요하다면 줄 끝에 잘 들어맞지 않는 단어를 잘라서 하이픈 처리 하기도 한다. 문단이 어떻게 조판되느냐는 문서 클래스에 따라 달라진다. 보통 많이 쓰는 방식은, 문단 첫 줄을 들여쓰기 하고, 문단과 문단 간격을 줄간격과 같이 잡아 더 넓게 벌리지 않는 것이다. 좀더 자세한 사항은 5.3.2를 참고하기 바란다.

가끔 LaTeX에게 줄을 바꾸거나 쪽을 나누라는 명령을 사용자가 내려야 하는 경우도 있다.

다음 명령은 줄을 바꾸어 새 줄로 시작하되 새로운 문단으로 시작하지는 말라는 명령이다.

`\\ or \newline`

다음 명령은 강제 줄바꿈을 하되 쪽 나누기는 방지하는 명령이다.

`\\*`

쪽을 나누어 새로운 쪽으로 시작하려면 다음과 같이 한다.

`\newpage`

줄나눔이나 쪽 나눔에 쓰이는 명령은 다음과 같다.

`\linebreak[`n`]`, `\nolinebreak[`n`]`, `\pagebreak[`n`]`, `\nopagebreak[`n`]`

이 명령의 동작은 이름이 가리키는 바대로다(줄 나누기, 줄 나누지 말기, 쪽 나누기, 쪽 나누지 말기). 사용자가 n 인수를 지정하여 명령의 동작 방식을 선택적으로 바꿀 수 있게 되어 있다. n 값으로는 0에서 4까지의 숫자가 올 수 있다. n을 4 미만으로 설정하면, LaTeX은 결과가 아주 나쁠 경우 사용자 명령을 무시할 수 있다. "break" 명령과 "new" 명령을 혼동하지 말기 바란다.[2] "break" 명령을 쓰더라도, LaTeX은 여전히 쪽의 오른쪽 끝과 쪽 전체 길이를 맞추려고 한다(다음 절에서 설명하듯이). 정말 "새로운 행"을 시작하고 싶으면, "줄 나누기"가 아니라 "새 줄로"를 의미하는 명령을 사용해야 할 것이다. 이름이 왜 다른지 생각해 보라!

[2] 여기서, linebreak는 "줄 나누기", newline은 "줄 바꾸기", 또는 "새 줄로"라고 번역하였다.[역자]

LaTeX은 항상 가능한 한 최선의 줄바꿈을 하려고 한다. LaTeX의 상당히 정밀한 기준을 충족시키는 줄바꿈 방법을 찾지 못하면, 한 행을 문단 폭 범위보다 오른쪽으로 나가도록 배치한다. 입력 파일을 처리하던 LaTeX은 이럴 경우 ("overfull hbox") 라는 경고를 보여준다. 이런 일이 일어나는 것은 대개 LaTeX이 단어를 나누어서 하이픈 처리할 적절한 위치를 찾지 못한 때이다.[3] `\sloppy` 명령을 쓰면 LaTeX의 처리기준을 낮출 수 있다. 이 명령은 단어 사이의 간격을 더 증가시켜서 오른쪽으로 밀려나는 행이 나타나지 않도록 만든다. 그러나 최종 출력물은 최선의 결과를 보여주지는 않을 것이다. 이 때는 ("underfull hbox") 라는 경고가 나타난다. 대부분의 경우, 결과물은 그리 썩 좋은 모양이 아니다. `\fussy` 명령은LaTeX을 원래의 작동방식으로 되돌린다.

2.2.2 하이픈

LaTeX은 필요하다면 언제나 단어를 하이픈 처리 한다. 하이픈 처리 알고리듬이 적절한 하이픈 삽입 위치를 찾지 못하는 경우, 다음과 같은 명령을 사용하여 TeX에게 예외 처리 방식을 알려줌으로써 해결할 수 있다.

`\hyphenation{`*word list*`}`

이 명령은 인자로 열거된 단어의 하이픈 삽입이 "-" 으로 표시된 곳에서만 일어나도록 한다. 일반 문자나 일반 문자로 간주되는 기호로 이루어진 단어만이 이 명령의 인자로 사용될 수 있다. 하이픈 처리 힌트는 하이퍼네이션 명령이 주어질 때 활성화되어 있는 언어에 맞추어서 저장된다. 무슨 말인가 하면, 하이퍼네이션 명령을 문서의 전처리부에 두게 되면 (이 때 활성화된 언어는 영어이므로) 영어의 하이픈 처리에 작용할 것이다. 이 명령을 `\begin{document}` 이후에 두면서 babel 등을 이용하여 특정 언어를 지원하도록 만들어 두었다면, 하이픈 처리 힌트는 babel을 통하여 활성화된 언어에서 작동할 것이다.

다음 보기와 같이 했을 때, "hyphenation" 의 하이픈 처리 방식은 "Hyphenation" 과 같고, "FORTRAN" 과 "Fortran", "fortran" 은 어떤 경우에도 하이픈 처리 되지 않을 것이다. 특수 문자나 기호 (심볼 문자) 는 인자로 올 수 없다.

보기 :

```
\hyphenation{FORTRAN Hy-phen-a-tion}
```

명령어 `\-`는 분절 하이픈을 단어에 삽입한다. 동시에 이것은 그 단어에서 하이픈 처리가 허용되는 유일한 위치가 된다. 이 명령은 특히 특수 문자 (예를 들면, 억양 표시 붙은 문자) 가 포함된 단어를 하이픈 처리할 때 유용하다. 특수 문자가 들어 있는 단어는 LaTeX이 자동 하이픈 처리를 하기 어렵기 때문이다.

[3] LaTeX이 이런 문제점 (Overfull hbox) 에 대해 경고를 내면서 위반행의 번호를 보여주었다 해도, 실제 그것이 어느 행인지 찾기는 쉽지 않다. `\documentclass` 명령에 `draft` 옵션을 쓰면 이런 행들 오른쪽 여백에 굵은 검은 선을 표시해준다.

```
I think this is: su\-per\-cal\-%
i\-frag\-i\-lis\-tic\-ex\-pi\-%
al\-i\-do\-cious
```

I think this is: supercalifragilisticexpialidocious

여러 단어들을 같은 줄에 두고 싶을 때는 다음 명령을 사용한다.

`\mbox{`*text*`}`

이렇게 하면 인자로 처리된 단어들이 어떤 경우에도 함께 묶여 있게 된다.

```
My phone number will change soon.
It will be \mbox{0116 291 2319}.

The parameter
\mbox{\emph{filename}} should
contain the name of the file.
```

My phone number will change soon. It will be 0116 291 2319.

The parameter *filename* should contain the name of the file.

`\fbox`는 `\mbox`와 비슷하지만, 내용 주위로 테두리선을 그려준다.

2.3 미리 정의된 문자열

이전의 몇몇 예에서, 특별한 텍스트 문자열을 식자하기 위해 사용되는 아주 간단한 LaTeX 명령 몇 가지를 이미 본 적이 있다.

명령어	사용예	설명
`\today`	2019년 5월 13일	사용하는 언어에서의 현재 날짜 표기
`\TeX`	TeX	가장 선호하는 조판기
`\LaTeX`	LaTeX	지금 우리가 배우고 있는 것
`\LaTeXe`	LaTeX 2$_\varepsilon$	LaTeX의 최신 버전

2.4 특수 문자와 기호 문자

2.4.1 따옴표

타자기나 보통 에디터에서 글을 쓸 때처럼 따옴표로 "를 사용하면 안된다. 출판용으로 쓰이는 따옴표는, 여는 따옴표와 닫는 따옴표가 다른 부호를 사용한다. LaTeX에서는 여는 따옴표 표시로 두 개의 `(grace accent)를 사용하고 닫는 따옴표로 두 개의 '(apostrophe)를 사용한다. 작은 따옴표의 경우에는 각각 한 번씩만 사용하면 된다.

```
``Please press the `x' key.''
```

“Please press the ‘x’ key.”

2.4.2 대시와 하이픈

LaTeX에는 네 종류의 대시가 있다. 그 중의 세 가지는 각각 대시를 몇 번씩 연이어 써서 얻을 수 있다. 네 번째 것은 사실은 대시가 아니다. 수학의 빼기(minus) 기호이다.

```
daughter-in-law, X-rated\\
pages 13--67\\
yes---or no? \\
$0$, $1$ and $-1$
```

daughter-in-law, X-rated
pages 13–67
yes—or no?
0, 1 and −1

세 가지 대시 각각의 이름은 다음과 같다.[4] '-' 하이픈, '–' 대시, '—' 긴 대시 그리고 '−' 빼기 부호.

2.4.3 틸데 (~)

Web 문서 주소에서 자주 볼 수 있는 문자가 틸데이다. LaTeX에서 이 문자를 이용하려면 \~를 쓰는 방법이 있다. 그러나 이 경우에는 ~ 가 원하는 결과를 보여주지 않는다. 다음과 같은 방법을 사용하자.[5]

```
http://www.rich.edu/\~{}bush \\
http://www.clever.edu/$\sim$demo
```

http://www.rich.edu/~bush
http://www.clever.edu/~demo

2.4.4 도(度) 기호(○)

온도, 각도 등의 도(度) 기호를 LaTeX에서는 어떻게 나타내는가?[6]

```
Its $-30\,^{\circ}\mathrm{C}$,
I will soon start to
super-conduct.
```

Its −30 °C, I will soon start to super-conduct.

[4] 한국어에서 이 부호들은 영문과 조금 다른 용례로 쓰인다. 우선, 한국어에서는 하이픈이 없다. 대시에 해당하는 문장부호들이 어떻게 사용되는가는 94 페이지 '역자의 보충'을 참고하라.[역자]

[5] 웹 주소를 식자하기 위해서는 `url` 패키지를 사용할 수 있다. 이 패키지를 사용하면 틸데 문자를 그대로 써도 잘 표현해준다.[역자]

[6] textcomp 패키지를 이용하면 이 기호를 보다 쉽게 넣을 수 있다. `\textdegree` 또는 `\textcelsius` 명령을 지원한다.[역자]

2.4.5 말줄임표 (…)

타자기에서는 쉼표나 마침표 간격이 한 글자 간격과 같다. 그러나 도서 인쇄에서 이 부호들은 더 짧은 폭을 갖고 앞 글자에 매우 가깝게 붙는다. 그러므로 "말줄임표"를 쓰려고 마침표를 세 개 연달아 찍으면 안된다. 이러한 점 세 개를 나타내기 위한 특별한 명령이 따로 존재한다.

`\ldots`

이 명령을 쓴 경우와 마침표를 연이어 찍은 경우를 비교해 보자.

```
Not like this ... but like this:\\
New York, Tokyo, Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, …

2.4.6 합자(ligature)

합자(ligature)란 영문 조판에서 어떤 문자들이 결합할 때 그것을 한 글자씩 차례로 식자하는 것이 아니라 특별한 기호 문자를 써서 조판하는 것을 가리키는 말이다. 다음 예는 합자를 사용한 경우와 글자를 차례로 나열한 경우를 비교한 것이다.

합자를 이용한 것: ff fi fl ffi …
합자를 이용하지 않은 것: ff fi fl ffi …

합자 기능을 이용하고 싶지 않은 경우에는 합쳐지는 두 글자 사이에 `\mbox{}`를 삽입하여 떼어놓으면 된다. 두 단어가 합쳐져서 하나의 단어가 된 경우에 이렇게 할 필요가 있을 수 있다.

```
Not shelfful\\
but shelf\mbox{}ful
```

Not shelfful
but shelfful

2.4.7 억양 표시와 특수 문자

LaTeX은 여러 언어의 억양 표시들와 특수 문자를 쓸 수 있도록 한다. 'o' 문자에 적용된 여러 가지 억양 표시들을 표 2.1에 보였다. 이 표시들은 다른 문자에도 마찬가지로 적용할 수 있다.

i와 j위에 억양표시를 붙이려면 이 두 문자 위의 점을 지워야 한다. 이것은 `\i`, `\j`라고 입력하면 된다.

```
H\^otel, na\"\i ve, \'el\‘eve,\\
sm\o rrebr\o d, !‘Se\~norita!,\\
Sch\"onbrunner Schlo\ss{}
Stra\ss e
```

Hôtel, naïve, élève,
smørrebrød, !‘Señorita!,
Schönbrunner Schloß Straße

표 2.1: 억양표시와 특수문자

ò	\‘o	ó	\’o	ô	\^o	õ	\~o
ō	\=o	ȯ	\.o	ö	\"o	ç	\c c
ǒ	\u o	ǒ	\v o	ő	\H o	ǫ	\c o
ọ	\d o	o̲	\b o	o͡o	\t oo		
œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
å	\aa	Å	\AA				
ø	\o	Ø	\O	ł	\l	Ł	\L
ı	\i	ȷ	\j	!‘	!‘	?‘	?‘

2.5 국제 언어 지원

영어가 아닌 다른 언어를 사용해야 한다면, 다음 두 가지 경우에 해당하는 LaTeX 설정을 적절히 바꾸어야 한다.

1. 모든 자동으로 생성되는 문자열[7]을 새로운 언어에 적당하도록 맞추어야 한다. 이 변환은 Johannes Braams씨가 만든 babel 패키지를 사용하면 여러 언어에서 쉽게 할 수 있다.
2. 새로운 언어의 하이픈 처리 규칙을 LaTeX에게 알려주어야 한다. LaTeX에게 하이픈 규칙을 알려주기 위해서는 약간 변칙적인 방법을 쓰는데, 그것은 새로운 하이픈 패턴이 작동하도록 포맷 파일을 다시 만드는 것이다. 이 점에 대해서는 *Local Guide* [4]에 더 자세한 정보가 나와 있어야 할 것이다.

자신의 시스템이 이미 적절히 설정되어 있다면, `\documentclass` 명령 뒤에 다음과 같이 babel 패키지 사용 명령을 추가하면 된다.

```
\usepackage[language]{babel}
```

또, 자신이 사용하는 시스템에서 지원하는 *language*가 Local Guide에 열

[7] 차례, 그림 차례, ⋯.

거되어 있어야 한다. Babel은 선택한 언어의 하이픈 처리 법칙을 자동적으로 활성화시킨다. 설령 현재 시스템에 설치된 LaTeX 포맷이 그 언어의 하이픈 처리 규칙을 지원하지 않더라도 babel은 작동한다. 그러나 하이픈 처리 기능은 꺼지는데, 그 결과 조판된 문서의 외양에는 좋지 않은 영향을 줄 수 있다.

babel은 몇 종류의 언어에 대해서 특수 문자 입력을 쉽게 하는 새로운 명령을 지원한다. 독일어의 경우를 예로 들면, 많은 움라우트(äöü) 문자가 쓰인다. babel를 사용하면, `\"o`라고 쓰지 않고 `"o`라고 입력하는 것만으로 ö를 얻을 수 있다.

어떤 컴퓨터 시스템은 키보드로 직접 특수 문자를 입력할 수 있도록 되어 있다. LaTeX도 이러한 문자들을 다룰 수 있다. 1994년 12월 LaTeX 2_ε가 발표된 이후, 몇 가지 문자 입력 인코딩 지원이 LaTeX 2_ε 기본 배포판에 포함되게 되었다. inputenc 패키지를 살펴 보기 바란다.

`\usepackage[`*encoding*`]{inputenc}`

이 패키지를 사용할 때는, 작성한 입력 파일을 다른 사람의 컴퓨터에서 읽을 수 없을지도 모른다는 사실에 주의해야 한다. 그 컴퓨터 시스템에서는 다른 인코딩 방식을 사용하기 때문이다. 한 예로, 독일어 움라우트 ä가 PC에서는 132로 인코딩되는 반면, ISO-LATIN 1 인코딩을 사용하는 몇몇 Unix시스템에서는 228로 인코딩된다. 따라서, 이 기능을 사용할 때는 주의를 요한다. 아래에 자신이 사용하는 시스템 유형에 따라 일반적으로 사용되는 인코딩 방식을 보였다.

운영체제	인코딩
Mac	`applemac`
Unix	`latin1`
Windows	`ansinew`
OS/2	`cp850`

폰트 인코딩은 다른 문제이다. 이것은 TeX 폰트 내의 어느 위치에 각 문자가 저장되지를 정의하는 것이다. 오리지널 컴퓨터 모던(Computer Modern) TeX 글꼴은 예전 7비트 아스키 문자 집합인 128개 문자만을 포함하고 있다. 억양 표시 붙은 문자가 필요할 때, TeX은 일반 문자와 억양 표시를 결합하여 문자를 만들어 낸다. 이런 방식은 보기에 좋은 출력물을 얻을 수 있게 하지만 억양 표시 붙은 문자가 사용된 단어에 대해 하이픈 자동 처리가 작동하지 못하는 결과를 초래한다.

다행히, 현재 대부분의 TeX 배포본은 한 벌의 EC 폰트를 포함하고 있다. 이 폰트는 Computer Modern 폰트와 비슷한 모양을 하고 있지만, 유럽 언어에서 사용되는 억양 표시 붙은 문자를 특수 문자 형태로 포함하고 있는 것이다. 이 폰트를 사용하면 비영어권 언어로 쓰여진 문서들에서 하이픈 처리 기능이 더 잘 작동하도록 만들 수 있다. EC 폰트를 쓰고 싶으면 문서의 전처리부(preamble)

에 fontenc 패키지 사용을 선언하여 활성화한다.

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

2.5.1 독일어 지원

LaTeX으로 독일어 문서를 만드는 사람들에게 몇 가지 힌트를 주려 한다. LaTeX의 독일어 지원은 다음과 같이 하면 된다.

```
\usepackage[german]{babel}
```

LaTeX 시스템이 적절하게 구축되었다면, 이 명령은 독일어 하이픈 처리를 가능하게 한다. 또 입력 텍스트를 자동으로 독일어로 변환한다. 예를 들면 Chapter(장)를 Kapitel로 변환한다. 이와 더불어 독일어 입력 파일을 신속히 만들 수 있도록 하는 새로운 명령어들을 포함하고 있기도 하다. 표 2.2를 참조하길 바란다.

표 2.2: 독일어 특수문자

"a	ä	"s	ß
"‘	„	"’	“
"<	«	">	»
\dq	”		

2.5.2 한국어 지원[8]

LaTeX에서 한국어를 사용하기 위해서는 세 가지 문제가 해결되어야 한다.

1. 한국어 입력 파일을 만들 수 있는 환경이 필요하다. 한국어 입력 파일은 plain text이어야 하지만 ASCII 범위 밖의 문자를 사용하기 때문에 예를 들면 KSX1001(완성형) 문자를 입력하고 저장할 수 있는 에디터와 한국어 환경이 제공되어야 한다. 이것은 운영체제의 국제화와 관련된 것으로,

[8]이 문서의 저자인 Tobias Oetiker 씨는 각 언어 번역자들이 위의 2.5.1 절을 자국 언어 지원을 위한 내용으로 교체해줄 것을 요구하였다. 그러나 우리나라 LaTeX 사용환경의 특성상 교체는 적절하다고 생각되지 않아서 위의 절을 그대로 두고 한국어 지원을 위한 소절을 하나 추가한다. 이 절은 이 문서의 한국어판 번역자를 대표하여 김강수가 집필하였다.

Windows 시스템의 경우 한글 Windows를 쓰면 되지만 다른 플랫폼에서는 적절한 한글 입력 방법이 이용가능해야 할 것이다.

2. 한국어 입력 파일을 처리할 수 있는 LaTeX 패키지가 있어야 한다. 현재 가장 많이 쓰이는 한국어 처리 패키지는 은광희 님이 만든 HLaTeX과, 차재춘 님이 만든 hLaTeXp, Werner Lemberg 씨가 유지하는 CJK 패키지가 있다. 이 가운데 HLaTeX과 hLaTeXp는 몇 가지 명령과 내부 처리루틴은 다르지만 완성형 한글을 처리하는 시스템이라는 점에서는 거의 동일하고, 다만 사용하는 폰트가 크게 차이가 난다. CJK 패키지는 완성형 입력 파일 이외에 UTF-8 유니코드 입력 파일도 처리할 수 있다는 점과, 일본어/중국어/한국어를 모두 쓸 수 있다는 장점이 있지만, 공개된 글꼴이 부족한 점이 한계라고 할 수 있다.[9]

3. 한국어 출력물을 만들 수 있는 글꼴이 있어야 한다. 위의 한국어 패키지들은 각각 몇 가지 한국어 폰트를 함께 제공한다. 그 가운데 HLaTeX은 UHC 글꼴(type 1)과 문화부 글꼴(truetype)을 포함하여 배포되고 있고, HLaTeX의 이전 버전 글꼴을 CJK에서 사용하고 있다. 이밖에 유니코드 글꼴인 cyberbit truetype 글꼴은 CJK 패키지와 함께 쓸 수 있고, 공개된 백묵 글꼴도 역시 `ttf2tfm`와 같은 유틸리티를 사용하면 HLaTeX이나 CJK 패키지에서 사용할 수 있다. 비교적 다양한 TTF 폰트를 LaTeX에서 사용할 수 있도록 하는 방법은 개발 중이다.

이 문서를 만든 HLaTeX의 경우에 한국어를 사용하려면, 전처리부에서 다음과 같이 선언한다.

```
\usepackage{hangul}
```

이 명령은 장, 절, 소절 및 목차, 그림 목차 등의 문자열을 모두 바꾸어주고, 자동 조사 처리 기능을 활성화한다. 만약, 자동 조사 처리와 같은 기능을 쓰지 않고 오직 한글을 식자만 하겠다면,

```
\usepackage{hfont}
```

를 쓰면 된다. 이제, 한글이 포함된 문서를 작성하여 컴파일해 볼 수 있다.

한국어 LaTeX 사용의 지침서는 HLaTeX의 경우, 함께 제공되는 문서인 *HLaTeX Guide*를 보라. 그밖에, 한국어 처리를 위한 지원이 "한글 TeX 사용자 그룹"(http://www.ktug.or.kr/)에서 이루어지고 있다. 특히 이 사이트에서

[9] 각 패키지를 찾아볼 수 있는 곳은 다음과 같다:
HLaTeX : `CTAN:/tex-archive/language/korean/HLaTeX/`
CJK : `CTAN:/tex-archive/language/korean/CJK/`
hLaTeXp : `http://knot.kaist.ac.kr/htex/`

는 Truetype 글꼴을 HLaTeX이나 CJK 패키지와 함께 쓰는 방법에 대한 조언도 얻을 수 있을 것이다.

2.6 단어 간 간격

출력물의 오른쪽 끝이 가지런하게 정렬되도록 하기 위해서, LaTeX은 단어와 단어 간 공백을 적절히 늘리거나 줄이거나 한다. 글을 보다 읽기 쉽게 하기 위해서 문장의 끝에 약간 더 넉넉한 공백을 삽입한다. LaTeX은 문장이 마침표나 물음표, 느낌표로 끝난다고 가정한다. 보통 대문자 다음에 오는 마침표는 약어를 표시하는 경우가 많으므로 마침표가 대문자 다음에 오면 약어를 표시하는 것으로 보고 문장이 끝난 것으로 간주하지 않는다.

LaTeX의 이런 가정에 맞지 않는 예외적인 경우라면 사용자가 스스로 그것을 명시해 주어야 한다. 백슬래시 다음에 공백 문자를 두면 폭이 일정한 공백을 만들어 낸다. 물결 표시(tilde) 문자 '~'는 줄바꿈이 일어나지 않는 고정폭 공백을 넣어준다. 마침표 앞에 `\@` 명령을 쓰면 대문자가 오더라도 문장의 끝나는 곳이라는 사실을 명시한다.

```
Mr.~Smith was happy to see her\\
cf.~Fig.~5\\
I like BASIC\@. What about you?
```

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

다음 명령을 쓰면 마침표 뒤에 더 넓은 공백을 넣는 것을 방지할 수 있다.

`\frenchspacing`

이것은 LaTeX에게 일반 문자의 다음보다 마침표 다음에 더 넓은 공백을 삽입하지 않도록 지시한다. 참고문헌을 쓸 때를 제외하고 비영어권 언어에서 이런 방식을 흔히 볼 수 있다. `\frenchspacing`을 사용하는 경우 `\@` 명령은 필요 없게 된다.

2.7 제목과 장, 절

독자가 글을 읽고 이해하기 쉽게 만들려면, 문서는 반드시 장(chapter)과 절(section), 그리고 소절(subsection) 등으로 나누어져 있어야 한다. LaTeX에서는 절 등의 제목을 인자로 취하는 특별한 명령으로 이것을 가능하게 한다. 이 명령을 올바른 순서로 사용하는 것은 전적으로 사용자의 책임이다.

`article` 클래스에서 사용할 수 있는 장/절 명령은 다음과 같다:

```
\section{...}           \paragraph{...}
\subsection{...}        \subparagraph{...}
\subsubsection{...}
```

또 report와 book 클래스에서는 다음 장/절 명령 두 개를 더 사용할 수 있다.

```
\part{...}              \chapter{...}
```

article 클래스에는 '장'이라는 개념이 없으므로, 이 article을 book에 하나의 장(chapter)처럼 삽입할 수도 있다. 절 제목과의 간격, 번호 붙이기, 제목의 글자 크기 등은 LaTeX이 알아서 처리해 준다.

장/절 명령 중 다음 두 가지는 약간 특별하다.

- \part 명령은 장의 번호 매기는 순서에 영향을 주지 않는다.
- \appendix 명령은 인자 없이 쓰인다. 다만 장의 번호 매기기를 숫자에서 문자로 바꾸어 준다.[10]

LaTeX은 문서를 마지막으로 컴파일할 때의 장/절 표제와 쪽 번호로부터 차례를 만든다. 다음 명령은 이 명령이 위치한 곳에 만들어진 차례를 넣는다.

`\tableofcontents`

새로운 문서의 올바른 차례를 얻기 위해서는 두 번 컴파일("LaTeX을 실행") 해야 한다. 어떨 때는 세 번 컴파일 해야 할 때도 있다. 그럴 경우 LaTeX이 컴파일이 더 필요하다는 것을 알려준다.

위에 언급된 모든 장/절 명령들은 "별표 붙은" 명령을 쓸 수 있다. "별표 붙은" 명령들이란 같은 명령 뒤에 *를 붙여서 사용하는 것을 말한다. 이 명령으로 만들어지는 장/절 표제는 차례에 나타나지도 않고 번호가 붙지도 않는다. 예를 들면 \section{Help} 명령에 이런 효과를 주고 싶으면 \section*{Help}처럼 쓴다.

보통 장/절 표제는 입력된 텍스트 그대로 차례에 나타난다. 그러나 표제가 너무 길어서 차례에 잘 들어맞지 않는 때에는 입력된 표제와는 다르게 차례에 표시할 수도 있다. 실제 표제 앞에 선택 인자를 써서 차례에 나타낼 짧은 제목을 지정하면 된다.

```
\chapter[Title for the table of contents]{A long
    and especially boring title, shown in the text}
```

[10] article에서는 절의 번호 매기기를 바꾼다.

문서 전체의 제목(title)을 붙이는 명령은 다음과 같다.

`\maketitle`

타이틀에 포함될 내용은 다음 명령에 의하여 지정되어 있어야 한다.

`\title{...}`, `\author{...}` 그리고 필요하다면 `\date{...}`

`\author` 명령의 인자 안에서는, `\and` 명령으로 여러 명의 저자 이름을 구분하여 쓸 수 있다.

9 쪽의 그림 1.3에서 위에 언급한 여러 가지 명령의 사용례를 보인 적이 있다.

위에서 설명한 장/절 명령 말고도, LaTeX 2ε는 `book` 클래스에서 쓰기 위한 명령 세 개를 더 갖추고 있다. 이것들은 책(출판물)을 몇 부분으로 나눌 때 쓰인다. 이 명령들은 보통 책에서 볼 수 있는 것처럼 장/절 제목 붙이기와 쪽번호 매기기가 각 부분별로 따로 이루어지도록 해준다.

`\frontmatter` 를 `\begin{document}` 바로 다음에 사용하면 쪽 번호가 로마숫자로 바뀐다. "별표 붙은" 절 표시 명령어(예를 들면, `\chapter*{Preface}`)를 사용하면 LaTeX이 절 번호를 붙여나가는 것을 막을 수 있다.

`\mainmatter` 는 책의 첫번째 장이 시작하기 바로 직전에 사용한다. 이 명령어는 쪽번호를 아라비아 숫자로 바꾸어주며 쪽번호 매기기를 새로 시작한다.

`\appendix` 는 책을 작성할 때 부록 부분의 시작점을 정의한다. 이 명령어 다음에 나오는 장(chapter)들은 숫자가 아니고 문자로 번호가 매겨진다.

`\backmatter` 는 참고문헌이나 색인과 같은 마지막 항목 직전에 사용되어야 한다. 표준적인 문서 클래스를 쓸 때 외관상 별다른 변화는 없다.

2.8 상호 참조

단행본, 보고서, 논문(기사)에서는 그림, 표 또는 텍스트의 특정 부분을 상호 참조(cross-reference)하는 경우가 많다. LaTeX은 상호 참조를 위한 다음 명령을 제공한다.

`\label{`*marker*`}`, `\ref{`*marker*`}`, `\pageref{`*marker*`}`

여기서 *marker*는 사용자가 적절히 선택하면 되는 식별자이다. LaTeX은 `\ref`를 만나면, 거기에 해당하는 `\label` 명령이 쓰인 곳을 찾아서, 장 · 절 · 그림 · 표 · 정리 등의 번호로 `\ref`를 바꾸어 넣는다. `\pageref`는 해당하는 `\label` 명령이

쓰인 쪽의 쪽수를 표시한다.[11] 장/절 표제를 달 때와 마찬가지로, 이들도 직전에 컴파일될 때 생성된 번호를 사용한다.

```
A reference to this subsection
\label{sec:this} looks like:
``see section~\ref{sec:this} on
page~\pageref{sec:this}.''
```

A reference to this subsection looks like: "see section 2.8 on page 32."

2.9 각주

각주를 붙이는 명령은 다음과 같다.

`\footnote{`*footnote text*`}`

각주는 현재 쪽의 하단에 인쇄된다. 각주는 항상 그 각주가 참조하는 단어나 문장의 뒤에 놓여야[12] 한다. 따라서, 한 문장 또는 문장의 일부를 참조하는 각주는 쉼표나 마침표 다음에 와야 한다.[13]

```
Footnotes\footnote{This is
  a footnote.} are often used
by people using \LaTeX.
```

Footnotes[a] are often used by people using LaTeX.

[a] This is a footnote.

[역자의 보충]

이 페이지의 각주에는 약간의 팁이 적용되었다. 본문에 설명이 없으므로 역자가 보충해둔다.

각주 13에는 다시 각주 표지(footnotemark)가 마지막에 붙어 있고, 마치 여기에 대한 주석인 것처럼 각주 14가 이어지고 있는데, 각주 안에서 다시 `\footnote` 명령을 쓸 수는 없으므로 `\footnote`로 붙인 것은 아니다. 각주 표지를 붙일 곳에 `\footnotemark`를 쓴다. 그리고 임의의 각주 본문은 `\footnotetext`로 써넣으면 된다. `\footnotemark`만을 쓰면 그 위치에 각주 표지만 붙고 `\footnotetext`를 쓰면 그 인자로 오는 텍스트가 각주 위치에 오게 된다. 이 명령들을 이용하면 같은 각주 번호를 두 번 이상 붙이면서 각주는 한 번 참조하게 할 수도 있다.

[11] 이 명령들은 그들이 무엇을 참조하고 있는지 모른다. 단지 자동적으로 생성된 마지막 번호를 저장하고 있을 뿐이다.

[12] 예를 들면 "놓이다"에 대한 각주는 이 위치에 와야 한다.

[13] 주의할 것은, 각주가 독자를 문서의 본문으로부터 벗어나게 한다는 사실이다. 각주를 읽고 있는 사람은 어쨌든 특이한 사람들이다. 그러므로, 말하고 싶은 것이 있다면 문서의 본문에 모두 포함시키는 것이 더 낫지 않겠는가.[14]

[14] "이정표가 가리키는 곳이 꼭 가야할 방향이어야 한다는 법은 없다." :-) (이 각주를 참조하는 각주 번호가 본문에는 없다는 것을 재미있게 표현한 말.[역자])

2.10 강조

원고를 타자로 칠 때는 중요한 단어를 강조하려면 밑줄을 긋는다.

`\underline{`*text*`}`

그러나 인쇄시에는 원고에 밑줄이 그어진 부분을 이탤릭 글꼴로 처리한다.[15] LaTeX이 제공하는 다음 명령은 텍스트를 강조 처리 하라는 것이다.

`\emph{`*text*`}`

이 명령이 실제로 어떻게 작용할 것인가는 전후 문맥에 의존한다. 다음 예에서 `\emph` 안의 `\emph`는 그 앞이 이탤릭 글꼴이므로 보통 글꼴로 바꾸어 강조를 나타내고 있다.

```
\emph{If you use
  emphasizing inside a piece
  of emphasized text, then
  \LaTeX{} uses the
  \emph{normal} font for
  emphasizing.}
```

If you use emphasizing inside a piece of emphasized text, then LaTeX uses the normal *font for emphasizing.*

즉, LaTeX에게 무언가를 강조하라고 하는 것과 다른 글꼴을 사용하라고 말하는 것에는 차이가 있다는 것을 알아야 한다.

```
\textit{You can also
  \emph{emphasize} text if
  it is set in italics,}
\textsf{in a
  \emph{sans-serif} font,}
\texttt{or in
  \emph{typewriter} style.}
```

You can also emphasize *text if it is set in italics,* in a *sans-serif* font, or in *typewriter* style.

2.11 환경

`\begin{`*environment*`}` *text* `\end{`*environment*`}`

여기서 *name*은 환경의 이름이다. 환경은 그 호출 순서가 지켜지는 한, 한 환경 안에서 다른 환경을 여러 번 호출할 수 있다.

[15] 한글 서적에서는 글꼴을 기울이기보다 돋움체 등으로 서체를 바꾸는 경우가 많다.[역주]

```
\begin{aaa}...\begin{bbb}...\end{bbb}...\end{aaa}
```

아래에서 중요한 환경들을 설명하겠다.

2.11.1 Itemize, Enumerate, Description 환경

`itemize` 환경은 단순 나열, `enumerate` 환경은 숫자 번호가 붙은 나열, `description` 환경은 사전에서 볼 수 있는 것 같은 설명을 위한 나열을 나타내는 데 쓰인다.

```
\flushleft
\begin{enumerate}
\item You can mix the list
environments to your taste:
\begin{itemize}
\item But it might start to
look silly.
\item[-] With a dash.
\end{itemize}
\item Therefore remember:
\begin{description}
\item[Stupid] things will not
become smart because they are
in a list.
\item[Smart] things, though, can be
presented beautifully in a list.
\end{description}
\end{enumerate}
```

1. You can mix the list environments to your taste:
   - But it might start to look silly.
   - With a dash.
2. Therefore remember:

   Stupid things will not become smart because they are in a list.

   Smart things, though, can be presented beautifully in a list.

2.11.2 Flushleft, Flushright, Center 환경

`flushleft`와 `flushright` 환경은 문단을 왼쪽 정렬시키거나 오른쪽 정렬시킨다. `center` 환경은 글을 가운데 정렬시킨다. 줄바꿈을 하기위해 \\를 써넣지 않으면 LaTeX이 스스로 줄바꿈을 결정한다.

```
\begin{flushleft}
This text is\\ left-aligned.
\LaTeX{} is not trying to make
each line the same length.
\end{flushleft}
```

This text is
left-aligned. LaTeX is not trying to make each line the same length.

```
\begin{flushright}
This text is right-\\aligned.
\LaTeX{} is not trying to make
each line the same length.
\end{flushright}
```

This text is right-
aligned. LaTeX is not trying to make each
line the same length.

```
\begin{center}
At the centre\\of the earth
\end{center}
```

At the centre
of the earth

2.11.3 Quote, Quotation, Verse 환경

quote 환경은 중요한 구절이나 예문을 인용하는 데 쓴다.

```
A typographical rule of thumb
for the line length is:
\begin{quote}
On average, no line should
be longer than 66 characters.
\end{quote}
This is why \LaTeX{} pages have
such large borders by default and
also why multicolumn print is
used in newspapers.
```

A typographical rule of thumb for the line length is:

> On average, no line should be longer than 66 characters.

This is why LaTeX pages have such large borders by default and also why multicolumn print is used in newspapers.

이와 유사한 환경이 두 개 더 있다. quotation와 verse 환경이 그것인데, quotation 환경은 문단 들여쓰기를 하기 때문에 여러 문단에 걸친 긴 인용문을 처리할 때 쓴다. verse 환경은 줄바꿈이 중요한 시(詩) 등을 쓸 때 유용하다. 각 행은 \\를 사용하여 나누어지고 각 연(verse) 뒤에는 빈 줄이 하나 따르게 된다.

```
I know only one English poem by
heart. It is about Humpty Dumpty.
\begin{flushleft}
\begin{verse}
Humpty Dumpty sat on a wall:\\
Humpty Dumpty had a great fall.\\
All the King's horses and all
the King's men\\
Couldn't put Humpty together
again.
\end{verse}
\end{flushleft}
```

I know only one English poem by heart. It is about Humpty Dumpty.

Humpty Dumpty sat on a wall:
Humpty Dumpty had a great
fall.
All the King's horses and all
the King's men
Couldn't put Humpty together
again.

2.11.4 그대로 보이기(Verbatim 환경)

\begin{verbatim}과 \end{verbatim} 사이에 오는 텍스트는 마치 타자를 친 것처럼, LaTeX 명령이 전혀 수행되지 않고 모든 줄바꿈과 공백들이 입력된 그대로 출력된다.

문단 안에서는, 다음 명령으로 비슷한 효과를 얻을 수 있다.

\verb+*text*+

+는 처음과 끝을 나타내는 문자의 한 예이다. *와 빈 칸을 제외한 다른 문자를 + 대신 사용할 수도 있다. 이 책에 보이는 많은 LaTeX 예들은 이 명령을 사용하여 조판된 것이다.

```
The \verb|\ldots| command \ldots

\begin{verbatim}
10 PRIM "HELLO WORLD ";
20 GOTO 10
\end{verbatim}
```

The \ldots command …

```
10 PRIM "HELLO WORLD ";
20 GOTO 10
```

```
\begin{verbatim*}
the starred version of
the      verbatim
environment emphasizes
the spaces   in the text
\end{verbatim*}
```

```
the␣starred␣version␣of
the␣␣␣␣␣␣␣verbatim
environment␣emphasizes
the␣spaces␣␣␣␣in␣the␣text
```

위에 별표 붙은 \verb 명령도 보였는데, 사용법은 비슷하다.

```
\verb*|like   this :-) |
```

```
like␣␣␣this␣:-)␣
```

verbatim 환경과 \verb 명령은 다른 명령의 인자 안에서 사용되면 안 된다.

2.11.5 Tabular 환경

tabular 환경은 보기좋은 표를 만드는 데 사용한다. 원한다면 가로줄이나 세로줄도 그릴 수 있다. LaTeX은 열(column)의 폭을 자동으로 결정한다.

\begin{tabular}{*table spec*}

이 명령에서, *table spec* 인자는 표의 모양을 지정한다. l은 각 열(column)을 왼쪽 정렬하고, r은 오른쪽 정렬한다. 가운데 정렬을 하려면 c를 사용한다.

`p{`*width*`}`를 쓰면 주어진 길이에 맞추어서 각 열 내의 텍스트를 정렬하고 자동 줄바꿈을 해준다. `|`는 세로줄을 나타낸다.

`tabular` 환경 내에서, `&`는 다음 열로 넘김, `\\`는 새로운 행의 시작, `\hline`는 가로줄 삽입에 쓰인다. `\cline{j-i}`를 이용하여서 부분적인 줄을 추가할 수도 있다. 여기에서 i와 j는 줄이 연장되는 곳의 열 번호이다.

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\ \cline{2-2}
11111000000 & binary \\
\hline \hline
1984 & decimal \\
\hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

```
\begin{tabular}{|p{4.7cm}|}
\hline
Welcome to Boxy's paragraph.
We sincerely hope you'll
all enjoy the show.\\
\hline
\end{tabular}
```

Welcome to Boxy's paragraph. We sincerely hope you'll all enjoy the show.

`@{}`. 열 분리자는 `@{...}` 형태로 구성을 지시할 수 있다. 이 명령은 열 사이의 공백을 없애고 소괄호 속에 있는 것(무엇이든 상관없다)으로 바꾸어 채운다. 이 명령의 일상적 용례 중의 하나를 다음 페이지에서 설명하는데, 이것은 소수점을 기준으로 한 숫자 정렬의 보기이다. 또 다른 응용 가능성은, 표 안에서 각 칸의 안쪽 여백을 없애는 것이다. `@{}`으로 지정하면 된다.

```
\begin{tabular}{@{} l @{}}
\hline
no leading space\\
\hline
\end{tabular}
```

no leading space

```
\begin{tabular}{l}
\hline
leading space left and right\\
\hline
\end{tabular}
```

leading space left and right

숫자로 된 열을 소수점에 맞추어 정렬하는 방법이 LaTeX 내부명령으로 되어 있지는 않으므로,[16] 약간의 트릭을 사용하여 구현해보려 한다. 즉, 두 개의 칼럼을 사용하되 왼쪽 칼럼은 오른쪽 정렬하고 오른쪽 칼럼은 왼쪽 정렬한 다음 칼럼 분리자를 소수점에 해당하는 점으로 처리하는 것이다. `\begin{tabular}`와 같은 줄에 쓴 `@{.}` 명령은 열 사이의 공백을 "."로 바꾸어주므로, 이것을 마치 소수점처럼 보이게 하는 것이다. 열 구분자(`&`)로 정수부와 나머지 소수부를 분리하는 것을 잊어 먹지 말기 바란다! 그런 다음, 칼럼의 표제는 `\multicolumn` 명령을 사용해서 방금 만든 숫자열 "칼럼"(사실은 두 개의 칼럼) 위에 하나의 칼럼인 것처럼 놓을 수 있다.

```
\begin{tabular}{c r @{.} l}
Pi expression       &
\multicolumn{2}{c}{Value} \\
\hline
$\pi$               & 3&1416  \\
$\pi^{\pi}$         & 36&46   \\
$(\pi^{\pi})^{\pi}$ & 80662&7 \\
\end{tabular}
```

Pi expression	Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

```
\begin{tabular}{|c|c|}
\hline
\multicolumn{2}{|c|}{Ene} \\
\hline
Mene & Muh! \\
\hline
\end{tabular}
```

Ene	
Mene	Muh!

tabular환경을 이용하여 조판작업을 할 경우에는 한 페이지안에 들어오도록 해야 한다. 만약에 상당히 긴 표를 만들고자 할 경우에는 supertabular과 longtabular 환경을 참조하는 것이 좋겠다.

2.12 떠다니는 표와 그림

요즘의 출판물들은 대부분 많은 그림과 표를 포함하고 있다. 이런 것들은 특별한 처리가 필요하다. 왜냐하면 쪽이 나누어져서는 안 되기 때문이다. 그림이나 표의 크기가 현재 쪽에 다 들어가지 않을 경우 생각해 볼 수 있는 방법은 표나 그림이 나오면 그 때마다 항상 새로운 쪽을 시작하는 것이다. 그러나 이렇게 하면 그 앞 페이지에 너무 많은 빈 공간이 남게 되고 이것은 보기에 좋지 않다.

[16] 만약 사용자의 시스템에 'tools' 패키지 꾸러미가 설치되어 있으면, dcolumn 패키지를 살펴보기 바란다.

이 문제의 해결책은 현재 페이지를 텍스트로 이어 채우면서, 이 페이지에 맞지 않는 그림이나 표는 '떠다니게' 하는 것이다. LaTeX은 이러한 떠다니는 개체에 해당하는 두 가지 환경을 제공한다. 하나는 표이고, 또 하나는 그림이다. 이 환경을 최대한 유용하게 사용하려면 LaTeX이 이 떠다니는 개체를 어떻게 조작하는지를 대강이라도 알고 있는 것이 중요하다. 그렇지 않으면, LaTeX은 이것을 원하는 위치에 놓지 않기 때문에 오히려 커다란 혼란에 빠질 수도 있다.

먼저, LaTeX이 이런 떠다니는 개체를 위해 제공하는 명령을 살펴보자.

`figure`나 `table` 환경으로 둘러 싸인 것은 무엇이든 떠다니는 개체로 간주된다. 두 환경은 모두 위치 지정 선택 인자(*placement specifier*)라 불리는 선택 인자를 지원한다.

`\begin{figure}`[*placement specifier*] 또는 `\begin{table}`[*placement specifier*]

이 선택 인자들은 떠다니는 개체가 이동할 수 있도록 허락된 위치를 LaTeX에게 알려준다. *placement specifier*는 개체가 놓일 수 있는 허용 범위의 첫글자를 따서 만들어졌다. 표 2.3을 보기 바란다.

표 2.3: 개체가 놓일 수 있는 허용 범위

선택 사항	개체가 놓일 수 있는 허용 범위 …
`h`	*here*의 약자. 이 명령이 사용된 바로 그 위치. 작은 개체에 유용하다.
`t`	*top*의 약자. 쪽의 윗부분.
`b`	*bottom*의 약자. 쪽의 아래 부분.
`p`	*page*의 약자. 그림이나 표만 모아둔 특별한 쪽에 위치.
`!`	개체의 위치를 제어하는 변수[a]를 거의 고려하지 않도록 함.

[a]한 페이지에 올 수 있는 개체의 최대값 같은 것들

Note: `0pt`와 `1.05em`는 TeX에서 사용하는 단위들이다. 82 쪽의 표 5.5에 좀더 자세히 나와있다.

표는 예를 들면 다음과 같은 줄로 시작한다.

```
\begin{table}[!hbp]
```

선택 인자 `[!hbp]`는 LaTeX이 표를 바로 여기 놓든지(`h`), 바닥에 놓든지(`b`) 혹은 특별한 페이지에 놓든지(`p`)하라는 것이고, 이렇게 할 때 설령 보기 좋지 않더라도 상관없다는(`!`) 것이다. 기본값으로 설정되어 있는 것은 `[tbp]`이다.

LaTeX은 각각의 떠다니는 개체를 모두 사용자가 정의한 대로 위치시킨다. 만약 현재 쪽에 개체가 올 수 없다면, 그림 대기열이나 표 대기열(queue)[17]로

[17]이것은 먼저 들어간 것이 먼저 나오는(fifo – 'first in first out') 큐(queue)이다.

가서 출력을 기다리게 된다. 새로운 쪽이 시작되면, LaTeX은 먼저 이 쪽이 대기열의 떠다니는 개체들로 채울 특별한 '플로트' 페이지가 될 수 있는지 여부를 확인한다. 만약 이것이 불가능하면, 각 대기열의 첫번째 개체를 텍스트의 현재 위치에서 나온 것으로 보고 처리하려 한다. 즉 LaTeX은 다시 그 개체의 위치 지정자('h'는 더이상 가능하지 않으므로 무시되지만)에 따라 놓으려고 해본다. 텍스트 중에 새로운 떠다니는 개체가 나오면 그것은 적절한 대기열로 보내진다. LaTeX은 각 그림과 표가 나타난 순서를 엄격하게 유지한다. 바로 이 이유 때문에 놓을 수 없는 그림 뒤에 나온 모든 그림들이 문서의 끝에 나타나는 것이다. 그러므로,

> 만약 LaTeX이 사용자가 원하는 대로 그림을 위치시키지 못한다면, 그것은 두개의 떠다니는 개체의 대기열 가운데 어떤 하나가 다른 하나를 방해하기 때문이다.

LaTeX에 단일위치지정자(single-location placement specifier)를 부여하는 것이 불가능하지 않지만, 이것은 문제를 일으킬 수 있다. 떠다니는 개체가 지정된 위치에 들어갈 수 없으면 꼼짝할 수 없게 되며, 다음에 등장할 떠다니는 개채들이 제자리에 찾아들어가는 것을 막는다. 특히 [h]옵션은 절대로 사용하면 안된다. 이러한 문제가 너무 심각하기 때문에 LaTeX 최근판에서는 자동적으로 [h]를 [ht]로 변환시킨다.

약간 어렵게 설명을 해왔는데, 아직 **표**와 **그림** 환경에 대하여 언급할 것이 조금 더 남아 있다.

`\caption{`*caption text*`}`

위의 명령을 사용하여, 그림이나 표의 제목(캡션)을 달 수 있다. 그림이나 표의 최종 번호와 "Figure(그림)" 또는 "Table(표)"이라는 문자열은 LaTeX이 자동적으로 붙여준다.

`\listoffigures` and `\listoftables`

이 두 명령은 `\tableofcontents` 명령과 비슷하게 작동하는데, 각각 그림과 표의 차례를 출력해 준다. 이 차례에는 그림과 표의 캡션이 전부 다시 나타날 것이다. 만약 긴 제목을 사용하게 된다면, 이 목차에 나타날 짧은 제목을 따로 만들어야 한다. `\caption` 명령 뒤의 대괄호 내에 원하는 짧은 제목을 써주면 된다.

```
\caption[Short]{LLLLLooooooonnnnnggggg}
```

`\label`과 `\ref`로 문서 내의 그림이나 표를 참조할 수 있다.

다음의 예는 정사각형을 하나 그린 후, 그것을 문서에 삽입하는 것이다. 이것은 완성된 문서 내에 어떤 그림을 집어넣을 공간을 미리 확보하고 싶을 때 유용하다.

```
Figure~\ref{white} is an example of Pop-Art.
\begin{figure}[!hbp]
\makebox[\textwidth]{\framebox[5cm]{\rule{0pt}{5cm}}}
\caption{Five by Five in Centimetres.} \label{white}
\end{figure}
```

위에 예에서, LaTeX이 그림을 바로 이 자리에(h)[18] 놓으려고 정말 어려운 (!) 시도를 할 것이다. 만약 이것이 불가능하면, 그림을 쪽의 바닥에 (b) 위치시키려 할 것이다. 그럼에도 불구하고 그림을 현재 쪽에 위치 시키는 것이 불가능하면, LaTeX은 이 그림(과 표 대기열에서 대기 중인 표들)을 담는 어떤 특별한 쪽에 위치시키는 것이 가능한지 여부를 결정할 것이다. 특별한 플로트 페이지를 만들기에는 그림이나 표가 충분치 않다면[19] LaTeX은 새로운 쪽을 만들면서 다시 한번 이 그림을 마치 텍스트의 바로 그 위치에서 생긴 것처럼 처리할 것이다.

어떤 경우 다음 명령들이 필요할지도 모른다.

`\clearpage` 혹은 `\cleardoublepage`

이것은 LaTeX으로 하여금 출력을 기다리고 있는 모든 개체들을 즉시 출력하고 새로운 쪽을 시작하게 한다. `\cleardoublepage`는 빈 쪽을 하나 만들어서라도 오른쪽 페이지로 가도록 한다.

LaTeX 2_ε 문서에 어떻게 PostScript 그림을 삽입하는지에 대해서는 이 책의 뒷부분에서 배우게 될 것이다.

2.13 풀리는 명령을 안 풀리게 하기

`\caption` 명령이나 `\section` 명령 등의 인자로 주어진 텍스트는 문서에서 한 번 이상 식자되어야 할 때가 있다. 예를 들면 문서 본문과 목차에서 두 번 나타나는 것처럼. 어떤 명령은 `\section`과 같은 장절 명령의 인자로 오게 되면 실행되지 않는 것이 있는데, 이것을 풀리는 명령이라 한다. 풀리는 명령으로 대표적인 것은 `\footnote`, `\phantom`과 같은 것이 있다. 이러한 풀리는 명령을 뜻한 대로 작동하게 하려면 풀리지 않도록 보호해주어야 한다(매번 그럴 필요는 없겠지만). 풀리는 명령 앞에 `\protect`를 써주면 된다.

[18] 기다리고 있는 그림 대기열이 비어 있다고 가정할 때

[19] 플로트 페이지를 만들기에 충분한가의 여부는 각 대기열에서 대기 중인 표와 그림을 모은 세로 길이가 `\floatpagefraction`이라는 상수값을 충족하는가의 여부에 달려 있다. 표준 LaTeX 클래스에서는 일반적으로 이 값이 0.5로 지정되어 있는데, `\renewcommand`로 사용자가 조절하는 것도 가능하다.[역자]

\protect는 그 명령 오른쪽에 오는 명령에만 효력이 있을 뿐이고 그 인자에는 영향을 미치지 않는다. 대부분의 경우 \protect를 남발해도 큰 문제는 없다.

```
\section{I am considerate
      \protect\footnote{and protect my footnotes}}
```

제3장

수식의 조판

자, 이 장에서는 TEX의 가장 큰 장점 하나를 다루어보자. 바로 수식 조판이다. 여기서 다루는 것은 대략적인 사용법에 불과하다는 걸 알아두자. 하지만 대다수 사용자에게 필요한 사항은 이 장에서 다룬 내용이면 충분하리라 본다. 원하는 수학식 표기 방법을 여기서 찾을 수 없다 해도 실망할 것은 없다. $\mathcal{AMS}$-LATEX[1] 패키지나 다른 패키지를 이용하면 대부분 해결할 수 있을 것이다.

3.1 개관

LATEX은 수식의 표현을 위해 "수식 모드"라는 특별한 모드를 사용한다. 문장 내에 포함되는 수식은 `\(`와 `\)`, `$`와 `$` 또는 `\begin{math}`와 `\end{math}` 사이에 넣어서 표현한다.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$c^{2}=a^{2}+b^{2}$
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach: $c^2 = a^2 + b^2$

```
\TeX{} is pronounced as
$\tau\epsilon\chi$.\\[6pt]
100~m$^{3}$ of water\\[6pt]
This comes from my $\heartsuit$
```

TEX is pronounced as $\tau\epsilon\chi$.

100 m^3 of water

This comes from my $\heartsuit$

더 큰 수식이나 공식은 줄마다 `$`을 사용하여 하나씩 조판하는 것보다 수식 보여주기(*display*) 하는 것이 더 나은데, 이 경우 `\[`와 `\]` 또는 `\begin{displaymath}`와

[1] `CTAN:/tex-archive/macros/latex/required/amslatex`

`\end{displaymath}`로 수식을 둘러싸면 된다. 이렇게 하면 수식의 오른쪽에 수식번호가 표시되지 않는다. 만약 수식번호를 보이게 하고 싶으면 `equation` 환경을 사용한다.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
\begin{displaymath}
c^{2}=a^{2}+b^{2}
\end{displaymath}
And just one more line.
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$c^2 = a^2 + b^2$$

And just one more line.

수식에 붙인 `\label`을 `\ref`로 참조하려면 다음과 같이 한다.

```
\begin{equation} \label{eq:eps}
\epsilon > 0
\end{equation}
From (\ref{eq:eps}), we gather
\ldots
```

$$\epsilon > 0 \tag{3.1}$$

From (3.1), we gather $\cdots$

수식 보이기(`displaymath`) 환경의 수식은 본문 중에 들어간 수식과 표현 방식이 다르다는 점에 주의하라.

```
$\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$
```

$\lim_{n \to \infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$

```
\begin{displaymath}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{displaymath}
```

$$\lim_{n \to \infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$$

수식 모드와 텍스트 모드 사이에는 다른 점이 있다. 예를 들면 수식 모드에서는,

1. 대부분의 빈 칸과 줄바꿈 문자들은 아무런 효과를 갖지 않는다. 수식의 빈 칸과 줄바꿈은 수식 표현 자체에서 논리적으로 나타나야 할 자리에 나타나거나, `\,`, `\quad` 혹은 `\qquad`과 같은 특수 명령어를 사용한 경우에 나타날 뿐이다.

2. 빈 줄은 허용하지 않는다. 즉 각 수식마다 한 개의 문단을 이루어야 한다.

3. 모든 글자는 변수 이름으로 간주되어 입력된 대로 식자된다. 수학식의 변수명은 기울인 글꼴을 사용하고 수식 자체의 독특한 간격을 스스로 결정하는데, 만약 수식 내에서 일반 텍스트(곧게 선 로만 글꼴에 일반 텍스트의 간격 주기를 적용한)를 식자하고자 하는 부분이 있으면 `\textrm{...}` 명령어를 사용해서 입력해야 한다.

```
\begin{equation}
\forall x \in \mathbf{R}:
\qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \qquad x^2 \geq 0 \tag{3.2}$$

```
\begin{equation}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbf{R}
\end{equation}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbf{R} \tag{3.3}$$

수학자는 기호(심볼)를 사용할 때 무척 세심하게 신경을 쓴다. 아래 예에서 보듯이 amsfonts 혹은 amssymb 패키지에서 `\mathbb` 명령을 사용하여 얻어지는 '블랙보드 볼드'(blackboard bold) 문자를 사용하는 것이 관행인 경우도 있다. 위의 마지막 보기를 블랙보드 볼드로 쓰면,

```
\begin{displaymath}
x^{2} \geq 0\qquad
\textrm{for all }x\in\mathbb{R}
\end{displaymath}
```

$$x^2 \geq 0 \qquad \text{for all } x \in \mathbb{R}$$

3.2 수식 모드의 그룹짓기

대부분의 수식 모드 명령어는 오직 명령어 다음에 오는 한 문자에 대해서만 효력을 갖는다. 만약 여러 문자에 대해 효력이 있게 하려면 중괄호 `{...}`를 사용하여 함께 묶으면 된다.

```
\begin{equation}
a^x+y \neq a^{x+y}
\end{equation}
```

$$a^x + y \neq a^{x+y} \tag{3.4}$$

3.3 수학 기호 및 수식의 표현

이 절에서는 수식 조판에서 가장 중요하게 쓰이는 명령 몇 가지를 설명한다. 수학 기호를 조판하는 데 쓰이는 명령을 57 쪽 3.10 절에 목록으로 작성해 두었으므로 참고하기 바란다.

그리스 소문자는 \alpha, \beta, \gamma, …와 같이 나타내며, **그리스 대문자**는 \Gamma, \Delta, …[2]와 같이 써넣는다.

```
$\lambda,\xi,\pi,\mu,\Phi,\Omega$
```

$\lambda, \xi, \pi, \mu, \Phi, \Omega$

윗첨자와 아래첨자는 각각 ^와 _ 문자를 사용하여 나타낸다.

```
$a_{1}$ \qquad $x^{2}$ \qquad
$e^{-\alpha t}$ \qquad
$a^{3}_{ij}$\\
$e^{x^2} \neq {e^x}^2$
```

$a_{1} \qquad x^{2} \qquad e^{-\alpha t} \qquad a^{3}_{ij}$
$e^{x^2} \neq {e^x}^2$

제곱근은 \sqrt로 입력하며, n-제곱근(n^{th} root)은 \sqrt[n]이라고 쓰면 만들어진다. 근호의 크기는 LaTeX이 자동으로 결정한다. 근호만 필요하다면 \surd를 사용한다.

```
$\sqrt{x}$ \qquad
$\sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$
```

$\sqrt{x} \qquad \sqrt{ x^{2}+\sqrt{y} }$
\qquad $\sqrt[3]{2}$\\[3pt]
$\surd[x^2 + y^2]$

\overline, \underline 명령은 지정된 부분의 위 아래에 **수평 줄**을 그린다.

```
$\overline{m+n}$
```

$\overline{m+n}$

\overbrace, \underbrace 명령은 지정된 범위에 걸치는 긴 **수평 중괄호**(horizontal braces)를 만든다.

```
$\underbrace{ a+b+\cdots+z }_{26}$
```

$\underbrace{ a+b+\cdots+z }_{26}$

작은 화살표나 물결표(tilde)와 같은 억양 표시를 변수에 첨가하려면, 57 쪽 표 3.1의 명령들을 사용한다. 여러 글자에 걸치는 넓은 모자형 심볼이나 큰 물결표는 \widetilde \widehat으로 만들 수 있다. ' 심볼은 프라임(prime)을 붙여준다.

[2] 대문자 알파(Alpha)는 로만 A와 모양이 동일하므로 LaTeX 2ε는 이것을 정의해두지 않았다. 새로운 수식 코딩 방법이 만들어지면 달라질 것이다.

```
\begin{displaymath}
y=x^{2}\qquad y'=2x\qquad y''=2
\end{displaymath}
```

$$y = x^2 \qquad y' = 2x \qquad y'' = 2$$

벡터는 일반적으로 변수 상단에 작은 화살표 기호를 붙이는데, 이것을 얻으려면 `\vec` 명령을 쓴다. `\overrightarrow`와 `\overleftarrow` 명령을 쓰면 A에서 B로 가는 벡터를 나타낼 수 있다.

```
\begin{displaymath}
\vec a\quad\overrightarrow{AB}
\end{displaymath}
```

$$\vec{a} \quad \overrightarrow{AB}$$

대부분의 경우 곱하기를 나타내기 위해 점을 찍을 필요는 없다. 그러나 가끔 독자들이 수식을 더 잘 판독하도록 하기 위해서 점을 찍어주면 좋은 경우가 있다. 이럴 경우에는 `\cdot`를 사용한다.

```
\begin{displaymath}
v = {\sigma}_1 \cdot {\sigma}_2
    {\tau}_1 \cdot {\tau}_2
\end{displaymath}
```

$$v = \sigma_1 \cdot \sigma_2 \tau_1 \cdot \tau_2$$

변수들은 기울어진 글자체로 나타내는 데 비해, log 같은 함수 이름은 곧게 선 (로만) 글자체로 쓰는 것이 일반적이다. LaTeX은 주요 함수 이름들을 조판하기 위하여 다음과 같은 명령을 제공한다.

```
\arccos  \cos   \csc  \exp   \ker     \limsup  \min
\arcsin  \cosh  \deg  \gcd   \lg      \ln       \Pr
\arctan  \cot   \det  \hom   \lim     \log      \sec
\arg     \coth  \dim  \inf   \liminf  \max      \sin
\sinh    \sup   \tan  \tanh
```

```
\[\lim_{x \rightarrow 0}
\frac{\sin x}{x}=1\]
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

나머지 연산(modulo) 함수에는 두 가지 명령이 있다. 이항 연산 "$a \bmod b$"는 `\bmod` 명령을 사용하여 표현하며, "$x \equiv a \pmod{b}$"와 같은 표현은 `\pmod` 명령을 쓴다.

분모 · 분자가 아래위로 붙는 보통 **분수 표현**은 `\frac{...}{...}` 명령으로 만든다. 가끔 1/2 처럼 슬래시로 분수를 나타내는 것이 더 낫다는 사람도 있는데, 서너 개 정도의 '분수'만 쓰는 경우 이쪽이 더 보기 좋기 때문이다.

```
$1\frac{1}{2}$~hours
\begin{displaymath}
\frac{ x^{2} }{ k+1 }\qquad
x^{ \frac{2}{k+1} }\qquad
x^{ 1/2 }
\end{displaymath}
```

$1\frac{1}{2}$ hours

$$\frac{x^2}{k+1} \qquad x^{\frac{2}{k+1}} \qquad x^{1/2}$$

상하 수식을 조판하려면 {... \choose ...} 또는 {... \atop ...} 명령을 쓸 수 있다. 후자는 전자와 결과가 동일하지만 아래위를 묶는 괄호가 붙지 않는다.[3]

```
\begin{displaymath}
{n \choose k}\qquad {x \atop y+2}
\end{displaymath}
```

$$\begin{displaymath}
{n \choose k}\qquad {x \atop y+2}
\end{displaymath}$$

이항관계(binary relations)를 나타내기 위해서는 기호 위에 기호를 써넣을 수 있으면 편하다. \stackrel 명령은 정상적으로 식자되는 두번째 인자의 위에 첫번째 인자를 첨자크기로 적어준다.

```
\begin{displaymath}
\int f_N(x) \stackrel{!}{=} 1
\end{displaymath}
```

$$\begin{displaymath}
\int f_N(x) \stackrel{!}{=} 1
\end{displaymath}$$

적분 기호는 \int 명령으로 생성할 수 있으며, **합**을 나타내는 기호는 \sum 으로 만들 수 있다. 적분이나 $\sum$의 상한과 하한 값은 윗첨자와 아래 첨자를 나타낼 때 쓰는 ^와 _으로 지정할 수 있다. [4]

```
\begin{displaymath}
\sum_{i=1}^{n} \qquad
\int_{0}^{\frac{\pi}{2}} \qquad
\prod_\epsilon
\end{displaymath}
```

$$\begin{displaymath}
\sum_{i=1}^{n} \qquad
\int_{0}^{\frac{\pi}{2}} \qquad
\prod_\epsilon
\end{displaymath}$$

괄호나 **시작과 끝을 나타내는 문자**는 TEX에서 정의된 심볼을 그대로 쓴다.(보기, $[\langle \| \updownarrow$). 둥근괄호(소괄호)와 각진괄호(대괄호)는 키보드에서 그대로 입력하면 된다. 중괄호는 \{으로 입력한다. 그밖의 다른 시작과 끝을 나타내는 문자들은 특별한 명령을 쓴다.(예. \updownarrow). 59 페이지의 표 3.8에 그 개요가 있으므로 참고하라.

[3] 여기서 설명한 옛날 방식의 명령은 amsmath 패키지에서는 쓰이지 않는다는 점에 주의하라. \bimon과 \genfrac 명령으로 바뀌었다. 두번째 지적한 \genfrac 명령은 관련된 구조를 가진 모든 명령의 수퍼세트로서, 다양하게 정의하여 사용할 수 있다. 예를 들면 \atop과 비슷한 명령을 다음과 같이 만들어서 쓰면 된다.

```
\newcommand{\newatop}[2]{\genfrac{}{}{0pt}{1}{#1}{#2}}.
```

[4] $\mathcal{AMS}$-LATEX에서는 여러 줄의 윗첨자/아래첨자를 지원한다.

```
\begin{displaymath}
{a,b,c}\neq\{a,b,c\}
\end{displaymath}
```

$$a,b,c \neq \{a,b,c\}$$

시작과 끝을 나타내는 문자를 쓸 때, 여는 문자 앞에 \left 명령을 두거나 닫는 문자 앞에 \right 명령을 두면, TEX이 알아서 여는 문자와 닫는 문자의 정확한 크기를 결정한다. \left와 \right는 반드시 짝을 이루어 사용되어야 하며, 둘 다 같은 줄에 조판되도록 해야 크기가 정확하게 결정된다는 것을 명심해야 한다. 열어놓고 닫지 않으려 한다면 닫는 괄호가 보이지 않게 하는 명령 '\right.'를 사용해야 한다.

```
\begin{displaymath}
1 + \left( \frac{1}{ 1-x^{2} }
    \right) ^3
\end{displaymath}
```

$$1 + \left(\frac{1}{ 1-x^{2} } \right) ^3$$

수식에서 시작과 끝을 나타내는 문자의 크기를 임의로 지정해야 하는 경우가 있다. \big, \Big, \bigg, \Bigg 명령을 각 문자의 앞에 놓으면 대부분 크기를 적당히 지정할 수 있다.[5]

```
$\Big( (x+1) (x-1) \Big) ^{2}$\\
$\big(\Big(\bigg(\Bigg($\quad
$\big\}\Big\}\bigg\}\Bigg\}$\quad
$\big\|\Big\|\bigg\|\Bigg\|$
```

$$\Big((x+1) (x-1) \Big) ^{2}$$
$$\big(\Big(\bigg(\Bigg(\quad \big\}\Big\}\bigg\}\Bigg\} \quad \big\|\Big\|\bigg\|\Bigg\|$$

수식 내에 **세 점**을 넣기 위한 명령이 몇 개 있다. \ldots는 점을 글자의 기준선(베이스라인)에 찍어주고,[6] \cdots는 문자박스의 중앙에 찍는다. 수직으로 세 점을 넣으려면 \vdots 명령어를 사용하며, 대각선 방향으로 점을 넣으려면 \ddots 명령을 쓰면 된다. 3.5 절에서 이에 대한 예제를 더 볼 수 있다.

```
\begin{displaymath}
x_{1},\ldots,x_{n} \qquad
x_{1}+\cdots+x_{n}
\end{displaymath}
```

$$x_1,\ldots,x_n \qquad x_1+\cdots+x_n$$

[5] 이 명령들은 글자 크기 변경 명령을 써도 원하는 대로 변화하지 않는다. 그리고 11pt, 12pt와 같은 옵션을 주더라도 역시 영향을 받지 않는다. 이 명령이나 옵션이 함께 작동하도록 하려면 exscale 또는 amsmath 패키지를 사용하라.

[6] "베이스라인(baseline)"에 대해서는 역자의 보충, 94 페이지를 참고할 것.[역자]

3.4 수식 모드의 간격

TEX이 설정하는 수식 내 간격이 맘에 들지 않는다면 특별한 간격주기 명령을 사용하여 조절할 수 있다. $\frac{3}{18}$ quad (␣)는 `\,`로 나타내며, $\frac{4}{18}$ quad (␣)는 `\:`로, $\frac{5}{18}$ quad (␣)는 `\;`로 나타낼 수 있다. 스페이스 문자(`\␣`)는 중간 정도의 간격으로 띄어주고, `\quad` (␣) 와 `\qquad` (␣) 는 넓게 간격을 벌려준다. `\quad` 간격은 사용 중인 글꼴에서 'M'자의 폭과 동일하다. `\!` 명령은 $-\frac{3}{18}$ quad ␣) 역방향으로 사이띄기(negative space)한다. 아래 예에서 보듯이 두 기호 사이의 간격을 줄여줄 때 쓸 수 있다.

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\int\!\!\!\int_{D} g(x,y)
  \, \ud x\, \ud y
\end{displaymath}
instead of
\begin{displaymath}
\int\int_{D} g(x,y)\ud x \ud y
\end{displaymath}
```

$$\iint_D g(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

instead of

$$\int\int_D g(x,y)\mathrm{d}x\mathrm{d}y$$

미분의 'd'기호는 관행상 로만 글자체로 식자한다.[7]

$\mathcal{AMS}$-LaTeX을 포함하면 쓸 수 있는 `\iint`, `\iiint`, `\iiiint`, `\idotsint` 명령은, 다중적분의 적분기호 사이 간격을 정밀하게 조정할 수 있는 다른 방법을 제공한다. amsmath 패키지 사용을 선언하였다면 위의 예는 아래와 같이 나타낼 수 있다.

```
\newcommand{\ud}{\mathrm{d}}
\begin{displaymath}
\iint_{D} \, \ud x \, \ud y
\end{displaymath}
```

$$\iint_D \mathrm{d}x\,\mathrm{d}y$$

좀 더 자세한 내용을 알려면 $\mathcal{AMS}$-LaTeX과 함께 배포되는 전자문서 testmath.tex를 보거나 "The LaTeX Companion" 제8장의 내용을 참조하면 된다.

3.5 수식의 수직 정렬

배열은 `array` 환경으로 조판한다. 이 array 환경은 `tabular`환경과 비슷하게 동작한다. 줄 바꿈은 `\\` 명령을 쓴다.

[7] 우리나라에서 만들어진 수학서적은 대부분 미분의 'd'도 기울어진 글꼴로 식자하는 경우가 많다.[역자]

```
\begin{displaymath}
\mathbf{X} =
\left( \begin{array}{ccc}
x_{11} & x_{12} & \ldots \\
x_{21} & x_{22} & \ldots \\
\vdots & \vdots & \ddots
\end{array} \right)
\end{displaymath}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

array 환경에서는 오른쪽 닫는 문자를 보이지 않게 하는 문자로 .을 쓸 수 있어서, 왼쪽에만 시작을 나타내는 문자를 크게 표시하고 오른쪽에는 쓰지 않으려는 경우에 사용할 수도 있다.

```
\begin{displaymath}
y = \left\{ \begin{array}{ll}
 a & \textrm{if $d>c$}\\
 b+x & \textrm{in the morning}\\
 l & \textrm{all day long}
  \end{array} \right.
\end{displaymath}
```

$$y = \left\{ \begin{array}{ll} a & \textrm{if } d > c \\ b + x & \textrm{in the morning} \\ l & \textrm{all day long} \end{array} \right.$$

tabular 환경 안에서와 같이 array 환경 안에서도 선을 그릴 수 있다. 예를 들면 행렬 원소를 구획하려면 다음과 같이 한다:

```
\begin{displaymath}
\left(\begin{array}{c|c}
 1 & 2 \\
\hline
3 & 4
\end{array}\right)
\end{displaymath}
```

$$\left(\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}\right)$$

여러 줄에 걸친 수식이나 수식군을 나타내기 위해, equation 환경 대신에 eqnarray와 eqnarray*를 사용할 수 있다. eqnarray를 사용하면 줄 마다 오른쪽에 수식 번호가 표시된다. eqnarray*에서는 번호가 어느 줄에도 나타나지 않는다.

eqnarray와 eqnarray* 환경은 {rcl} 형식의 세 칸(column) 짜리 표처럼 동작한다. 이들 세 개의 칸 중 가운데 칸은 등호나 부등호 기호를 나타내는 데 사용된다. 혹은 사용자가 원하는 기호를 사용하여 표현할 수도 있다. \\ 명령어는 줄바꿈을 나타낸다.

```
\begin{eqnarray}
f(x) & = & \cos x      \\
f'(x) & = & -\sin x    \\
\int_{0}^{x} f(y)dy &
 = & \sin x
\end{eqnarray}
```

$$f(x) = \cos x \tag{3.5}$$
$$f'(x) = -\sin x \tag{3.6}$$
$$\int_0^x f(y)dy = \sin x \tag{3.7}$$

등호의 좌우측 간격은 다소 넓다.
이것은 다음 예제에서 쓰인 것처럼, `\setlength\arraycolsep{2pt}`라고 설정하여 줄일 수 있다.

긴 수식은 자동으로 줄바꿈이 되지 않으므로 사용자는 이런 긴 식의 줄바꿈할 부분과 들여쓰기할 부분을 지정해야 한다. 제일 많이 쓰이는 두 가지 방법을 아래 예제에서 보였다.

```
{\setlength\arraycolsep{2pt}
\begin{eqnarray}
\sin x & = & x -\frac{x^{3}}{3!}
     +\frac{x^{5}}{5!}-{}
                     \nonumber\\
 & & {}-\frac{x^{7}}{7!}+{}\cdots
\end{eqnarray}}
```

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \\ - \frac{x^7}{7!} + \cdots \tag{3.8}$$

```
\begin{eqnarray}
\lefteqn{ \cos x = 1
     -\frac{x^{2}}{2!} +{} }
                     \nonumber\\
 & & {}+\frac{x^{4}}{4!}
     -\frac{x^{6}}{6!}+{}\cdots
\end{eqnarray}
```

$$\cos x = 1 - \frac{x^2}{2!} + \\ + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \tag{3.9}$$

`\nonumber` 명령은 수식의 오른쪽에 수식 번호를 넣지 않도록 한다.

위에 설명한 방법으로는 수식을 수직 정렬하여 가지런하게 배치하기 어려울 수 있다. amsmath 패키지는 좀더 강력한 표현 방법을 제공한다. (`split` 과 `align` 환경을 참고하라.)

3.6 도깨비글자 (Phantom)

도깨비를 본 적은 없지만 여전히 많은 사람의 마음에 한 자리를 차지하고 있다. LaTeX에서도 다를 것이 없다. phantom 문자를 사용하면 눈에 보이지는 않지만 자리를 차지하기 때문에 간격 문제를 교묘하게 처리하는 재미있는 방법으로 활용할 수 있다.

^와 _를 이용해서 상하첨자를 쓰는 경우 첨자의 세로 위치를 정렬하려 할 때 LaTeX은 이따금 너무 엄격해서 자유로운 간격 조절을 허용하지 않는 경우가 있다. 이 때 \phantom 명령을 사용하면 최종 출력물에는 나타나지 않는 문자만큼의 공간을 확보할 수 있다. 다음 예를 잘 살펴보면 도움이 될 것이다.

```
\begin{displaymath}
{}^{12}_{\phantom{1}6}\textrm{C}
\qquad \textrm{versus} \qquad
{}^{12}_{6}\textrm{C}
\end{displaymath}
```

$$ {}^{12}_{6}\textrm{C} \qquad \textrm{versus} \qquad {}^{12}_{6}\textrm{C} $$

```
\begin{displaymath}
\Gamma_{ij}^{\phantom{ij}k}
\qquad \textrm{versus} \qquad
\Gamma_{ij}^{k}
\end{displaymath}
```

$$ \Gamma_{ij}^{k} \qquad \textrm{versus} \qquad \Gamma_{ij}^{k} $$

3.7 수학 글꼴의 크기

수식 모드에서는 TeX이 전후 맥락에 맞추어 가장 적절한 글꼴 크기를 선택한다. 예를 들면 윗첨자로 들어가는 문자는 작은 글자체로 조판한다. 만약 로만 글자체로 수식의 일부분을 작성하고자 한다면, \textrm 명령을 사용하면 안 된다. 왜냐하면 글꼴 크기 조절 메커니즘이 작동하지 않기 때문이다. \textrm 명령은 일시적으로 텍스트 모드로 빠져나가게 하는 것이다. 수식 모드에서 글꼴 크기 조절 메커니즘이 동작하도록 하려면 \mathrm을 써야 하는데, 약간의 주의가 필요하다. \mathrm은 짧은 수식항목에서만 잘 동작할 것이다. 공백은 여전히 조절되지 않을 것이고, 억양 표시 붙은 문자들은 작동하지 않을 것이다.[8]

```
\begin{equation}
2^{\textrm{nd}} \quad
2^{\mathrm{nd}}
\end{equation}
```

$$ 2^{\textrm{nd}} \quad 2^{\mathrm{nd}} \tag{3.10} $$

TeX이 자동으로 크기를 선택하기는 하지만, 간혹 LaTeX에게 글자 크기를 지정해주어야 할 필요가 있는 경우도 있다. 수식 모드에서, 글꼴 크기는 네 가지 명령으로 지시한다.

\displaystyle (123), \textstyle (123), \scriptstyle (123) and \scriptscriptstyle (123).

[8] The $\mathcal{AMS}$-LaTeX 패키지는 \textrm 명령어와 크기 조절이 함께 작동하게 해준다.

스타일을 바꾸면 상 하한값의 표시방식에도 영향을 미친다.

```
\begin{displaymath}
\mathop{\mathrm{corr}}(X,Y)=
 \frac{\displaystyle
   \sum_{i=1}^n(x_i-\overline x)
   (y_i-\overline y)}
  {\displaystyle\biggl[
 \sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
```

$$\mathop{\mathrm{corr}}(X,Y)=\frac{\displaystyle\sum_{i=1}^n(x_i-\overline x)(y_i-\overline y)}{\displaystyle\biggl[\sum_{i=1}^n(x_i-\overline x)^2\sum_{i=1}^n(y_i-\overline y)^2\biggr]^{1/2}}$$

이 예제에서는 기본 `\left[ \right]`이 만드는 것보다 더 큰 각진괄호(대괄호)가 사용되었다.

3.8 정리(theorem), 법칙, ···

수학 문서를 작성하다 보면, 종종 "보조 정리(Lemmas)", "정의(Definitions)", "공리(Axiom)" 내지 이와 유사한 구조를 조판해야 할 필요가 생긴다. LaTeX이 지원하는 이에 관한 명령은 다음과 같다.

`\newtheorem{`*name*`}[`*counter*`]{`*text*`}[`*section*`]`

여기서 *name* 인자는 "정리(theorem)"를 구분하기 위해 붙이는 짧은 명칭(식별자)이다. *text* 인자는 최종 출력물에서 인쇄될 "정리(theorem)"의 실제 이름을 지정한다.

대괄호내의 인자는 옵션이다. 위의 두 대괄호는 "정리(theorem)"에 붙일 번호 매기기에 사용된다. *counter* 인자에는 앞에서 사용된 "정리(theorem)"의 *name*을 기입할 수 있다. 그렇게 하면 새로운 "정리"는 앞의 "정리"에서부터 번호가 연속되어 붙는다. *section* 지시어는 "정리"에 번호를 붙일 때 정리가 들어가기를 원하는 절(section) 번호를 함께 표시하도록 한다.

`\newtheorem` 명령이 TeX 문서의 전처리부에서 실행된 후라면, 문서 본문에서 다음 명령을 사용할 수 있다.

```
\begin{name}[text]
This is my interesting theorem
\end{name}
```

위의 예로써 충분히 이해하고 쓸 수 있을 것이다. 그밖의 의문점들을 해소하고 너무 복잡해 보이는 `\newtheorem` 환경에 대한 이해를 명확하게 했으면 하는 바람에서 다음 몇 가지 예를 보이겠다.

```
% definitions for the document
% preamble
\newtheorem{law}{Law}
\newtheorem{jury}[law]{Jury}
%in the document
\begin{law} \label{law:box}
Don't hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law~\ref{law:box}\end{jury}
\begin{law}No, No, No\end{law}
```

Law 1 *Don't hide in the witness box*

Jury 2 (The Twelve) *It could be you! So beware and see law 1*

Law 3 *No, No, No*

"Jury" 정리는 "Law" 정리와 동일한 카운터를 사용한다. 그러므로 다른 "Law" 정리들과 함께 일련번호가 붙게 된다. 괄호(대괄호) 안의 인자는 정리의 제목이나 참조사항 등을 써넣는 데 이용된다.

```
\flushleft
\newtheorem{mur}{Murphy}[section]
\begin{mur}
If there are two or more
ways to do something, and
one of those ways can result
in a catastrophe, then
someone will do it.\end{mur}
```

Murphy 3.8.1 *If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.*

"Murphy" 정리에 붙는 번호는 현재의 절(section) 번호와 연동된다. 필요하다면 다른 단위들, 예를 들어 장(chapter)이나 소절(subsection)에 대해서도 마찬가지로 할 수 있다.

3.9 굵은 기호 문자

LaTeX에서는 두꺼운 기호(볼드 심볼)을 얻기가 상당히 어렵다. 아마 아마추어 사용자가 이것을 남용하는 경향이 있어 의도적으로 그렇게 해 놓은 것 같다. 글꼴 바꿈 명령인 `\mathbf`는 두꺼운 글자를 만들지만, 이것은 로만 글꼴(바로 선 글씨체)로 나오게 된다. 하지만 수학 기호 문자는 일반적으로 이탤릭(기울인) 글꼴이 대부분이다. `\boldmath`란 명령이 있으나, 이것은 수식 모드를 빠져나가서 써야 한다. 기호 문자에도 영향을 미친다.

```
\begin{displaymath}
\mu, M \qquad \mathbf{M} \qquad
\mbox{\boldmath $\mu, M$}
\end{displaymath}
```

$$\mu, M \qquad \mathbf{M} \qquad \boldsymbol{\mu, M}$$

그래서 이 예에서는 쉼표(콤마)가 두꺼운 글자로 나오게 되었다. 그러나 이것은 원하는 바가 아니다.

amsmath에 포함되어 있는 amsbsy 패키지를 쓰거나 tools 꾸러미에 들어 있는 bm 패키지를 쓰면 두꺼운 기호 문제를 훨씬 쉽게 해결할 수 있는데, 다음과 같이 `\boldsymbol` 명령을 사용하면 된다.

```
\begin{displaymath}
\mu, M \qquad
\boldsymbol{\mu}, \boldsymbol{M}
\end{displaymath}
```

$$\mu, M \qquad \boldsymbol{\mu}, \boldsymbol{M}$$

3.10 수식 기호문자

수식 모드에서 일반적으로 사용할 수 있는 기호 문자(심볼)들을 모아서 다음에 몇 개의 표로 정리하였다.

표 3.12–3.16[9]의 기호문자(심볼)를 사용하기 위해서는 amssymb 패키지가 전처리부에서 포함되어야 하며, AMS 수식 글꼴이 시스템에 설치되어 있어야 한다. AMS 패키지와 글꼴이 설치되어 있지 않다면, `CTAN:/tex-archive/macros/latex/required/amslatex`를 참조하라.

표 3.1: 수학 모드의 억양 표시 붙은 글자

$\hat{a}$	`\hat{a}`	$\check{a}$	`\check{a}`	$\tilde{a}$	`\tilde{a}`	$\acute{a}$	`\acute{a}`
$\grave{a}$	`\grave{a}`	$\dot{a}$	`\dot{a}`	$\ddot{a}$	`\ddot{a}`	$\breve{a}$	`\breve{a}`
$\bar{a}$	`\bar{a}`	$\vec{a}$	`\vec{a}`	$\widehat{A}$	`\widehat{A}`	$\widetilde{A}$	`\widetilde{A}`

표 3.2: 그리스 소문자

α	`\alpha`	θ	`\theta`	o	`o`	υ	`\upsilon`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	ϕ	`\phi`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	φ	`\varphi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	χ	`\chi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	ψ	`\psi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ω	`\omega`
ζ	`\zeta`	ν	`\nu`	ς	`\varsigma`		
η	`\eta`	ξ	`\xi`	τ	`\tau`		

표 3.3: 그리스 대문자

Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

[9]이 표들은 David Carlisle이 처음 작성한 후 Josef Tkadlec이 제안한 바를 반영하여 확장한 `symbols.tex` 문서에서 얻은 것이다.

표 3.4: 이항 관계 연산 기호

아래 심볼은 \not 명령어를 이용해서 부정 관계 기호로 바꿀 수 있다.

$<$	<	$>$	>	$=$	=
$\leq$	\leq or \le	$\geq$	\geq or \ge	$\equiv$	\equiv
$\ll$	\ll	$\gg$	\gg	$\doteq$	\doteq
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong
$\sqsubset$ [a]	\sqsubset [a]	$\sqsupset$ [a]	\sqsupset [a]	$\bowtie$ [a]	\Join [a]
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\bowtie$	\bowtie
$\in$	\in	$\ni$	\ni , \owns	$\propto$	\propto
$\vdash$	\vdash	$\dashv$	\dashv	$\models$	\models
$\mid$	\mid	$\parallel$	\parallel	$\perp$	\perp
$\smile$	\smile	$\frown$	\frown	$\asymp$	\asymp
$:$	:	$\notin$	\notin	$\neq$	\neq or \ne

[a] 이 기호를 사용하려면 latexsym 패키지가 포함되어야 한다.

표 3.5: 이항 연산 기호

$+$	+	$-$	-		
$\pm$	\pm	$\mp$	\mp	$\triangleleft$	\triangleleft
$\cdot$	\cdot	$\div$	\div	$\triangleright$	\triangleright
$\times$	\times	$\setminus$	\setminus	$\star$	\star
$\cup$	\cup	$\cap$	\cap	$*$	\ast
$\sqcup$	\sqcup	$\sqcap$	\sqcap	$\circ$	\circ
$\vee$	\vee , \lor	$\wedge$	\wedge , \land	$\bullet$	\bullet
$\oplus$	\oplus	$\ominus$	\ominus	$\diamond$	\diamond
$\odot$	\odot	$\oslash$	\oslash	$\uplus$	\uplus
$\otimes$	\otimes	$\bigcirc$	\bigcirc	$\amalg$	\amalg
$\bigtriangleup$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown	$\dagger$	\dagger
$\lhd$ [a]	\lhd [a]	$\rhd$ [a]	\rhd [a]	$\ddagger$	\ddagger
$\unlhd$ [a]	\unlhd [a]	$\unrhd$ [a]	\unrhd [a]	$\wr$	\wr

표 3.6: 큰 기호

$\sum$	\sum	$\bigcup$	\bigcup	$\bigvee$	\bigvee	$\bigoplus$	\bigoplus
$\prod$	\prod	$\bigcap$	\bigcap	$\bigwedge$	\bigwedge	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup			$\bigodot$	\bigodot
$\int$	\int	$\oint$	\oint			$\biguplus$	\biguplus

표 3.7: 화살표

$\leftarrow$	\leftarrow or \gets	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\rightarrow$	\rightarrow or \to	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\iff$	\iff (bigger spaces)	$\leadsto$ [a]	\leadsto [a]

[a] 이 기호를 사용하기 위해서는 latexsym 패키지를 포함해야 한다.

표 3.8: 시작과 끝을 나타내는 기호

(	(	)	)	$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
[	[or \lbrack	]	] or \rbrack	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
{	\{ or \lbrace	}	\} or \rbrace	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\langle$	\langle	$\rangle$	\rangle	$\vert$	| or \vert	$\Vert$	\\| or \Vert
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil
/	/	\	\backslash		. (dual. empty)		

표 3.9: 시작과 끝을 나타내는 큰 기호

$\lgroup$	\lgroup	$\rgroup$	\rgroup	$\lmoustache$	\lmoustache	$\rmoustache$	\rmoustache
$\arrowvert$	\arrowvert	$\Arrowvert$	\Arrowvert	$\bracevert$	\bracevert		

표 3.10: 기타 기호 문자

…	\dots	⋯	\cdots	⋮	\vdots	⋱	\ddots
$\hbar$	\hbar	$\imath$	\imath	$\jmath$	\jmath	ℓ	\ell
$\Re$	\Re	$\Im$	\Im	$\aleph$	\aleph	$\wp$	\wp
$\forall$	\forall	$\exists$	\exists	$\mho$ [a]	\mho [a]	∂	\partial
$'$	'	$\prime$	\prime	$\emptyset$	\emptyset	∞	\infty
∇	\nabla	$\triangle$	\triangle	$\Box$ [a]	\Box [a]	$\Diamond$ [a]	\Diamond [a]
$\bot$	\bot	$\top$	\top	$\angle$	\angle	$\surd$	\surd
$\diamondsuit$	\diamondsuit	$\heartsuit$	\heartsuit	$\clubsuit$	\clubsuit	$\spadesuit$	\spadesuit
$\neg$	\neg or \lnot	$\flat$	\flat	$\natural$	\natural	$\sharp$	\sharp

[a] 이 기호를 사용하려면 latexsym 패키지가 포함되어야 한다.

표 3.11: 수학과 관계없는 기호

다음 기호들은 텍스트 모드에서도 사용할 수 있다.

†	\dag	§	\S	©	\copyright
‡	\ddag	¶	\P	£	\pounds

표 3.12: AMS의 시작과 끝을 나타내는 기호

⌜	\ulcorner	⌝	\urcorner	⌞	\llcorner	⌟	\lrcorner
|	\lvert	|	\rvert	‖	\lVert	‖	\rVert

표 3.13: AMS 그리스 문자와 히브리 문자

F	\digamma	$\varkappa$	\varkappa	ב	\beth	ד	\daleth	ג	\gimel

표 3.14: AMS 이항 관계 연산 기호

$\lessdot$	\lessdot	$\gtrdot$	\gtrdot	$\doteqdot$	\doteqdot or \Doteq
$\leqslant$	\leqslant	$\geqslant$	\geqslant	$\risingdotseq$	\risingdotseq
$\eqslantless$	\eqslantless	$\eqslantgtr$	\eqslantgtr	$\fallingdotseq$	\fallingdotseq
$\leqq$	\leqq	$\geqq$	\geqq	$\eqcirc$	\eqcirc
$\lll$	\lll or \llless	$\ggg$	\ggg or \gggtr	$\circeq$	\circeq
$\lesssim$	\lesssim	$\gtrsim$	\gtrsim	$\triangleq$	\triangleq
$\lessapprox$	\lessapprox	$\gtrapprox$	\gtrapprox	$\bumpeq$	\bumpeq
$\lessgtr$	\lessgtr	$\gtrless$	\gtrless	$\Bumpeq$	\Bumpeq
$\lesseqgtr$	\lesseqgtr	$\gtreqless$	\gtreqless	$\thicksim$	\thicksim
$\lesseqqgtr$	\lesseqqgtr	$\gtreqqless$	\gtreqqless	$\thickapprox$	\thickapprox
$\preccurlyeq$	\preccurlyeq	$\succcurlyeq$	\succcurlyeq	$\approxeq$	\approxeq
$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backsim$	\backsim
$\precsim$	\precsim	$\succsim$	\succsim	$\backsimeq$	\backsimeq
$\precapprox$	\precapprox	$\succapprox$	\succapprox	$\vDash$	\vDash
$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq	$\Vdash$	\Vdash
$\Subset$	\Subset	$\Supset$	\Supset	$\Vvdash$	\Vvdash
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\backepsilon$	\backepsilon
$\therefore$	\therefore	$\because$	\because	$\varpropto$	\varpropto
$\shortmid$	\shortmid	$\shortparallel$	\shortparallel	$\between$	\between
$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\pitchfork$	\pitchfork
$\vartriangleleft$	\vartriangleleft	$\vartriangleright$	\vartriangleright	$\blacktriangleleft$	\blacktriangleleft
$\trianglelefteq$	\trianglelefteq	$\trianglerighteq$	\trianglerighteq	$\blacktriangleright$	\blacktriangleright

표 3.15: AMS 화살표

⇠	\dashleftarrow	⇢	\dashrightarrow	⊸	\multimap
⇇	\leftleftarrows	⇉	\rightrightarrows	⇈	\upuparrows
⇆	\leftrightarrows	⇄	\rightleftarrows	⇊	\downdownarrows
⇚	\Lleftarrow	⇛	\Rrightarrow	↿	\upharpoonleft
↞	\twoheadleftarrow	↠	\twoheadrightarrow	↾	\upharpoonright
↢	\leftarrowtail	↣	\rightarrowtail	⇂	\downharpoonleft
⇋	\leftrightharpoons	⇌	\rightleftharpoons	⇃	\downharpoonright
↰	\Lsh	↱	\Rsh	⇝	\rightsquigarrow
↫	\looparrowleft	↬	\looparrowright	↭	\leftrightsquigarrow
↶	\curvearrowleft	↷	\curvearrowright		
↺	\circlearrowleft	↻	\circlearrowright		

표 3.16: AMS 부정 관계 연산 기호와 화살표

≮	\nless	≯	\ngtr	⫋	\varsubsetneqq
≨	\lneq	≩	\gneq	⫌	\varsupsetneqq
≰	\nleq	≱	\ngeq	⫅̸	\nsubseteqq
≰	\nleqslant	≱	\ngeqslant	⫆̸	\nsupseteqq
≨	\lneqq	≩	\gneqq	∤	\nmid
≨	\lvertneqq	≩	\gvertneqq	∦	\nparallel
≰	\nleqq	≱	\ngeqq	∤	\nshortmid
≴	\lnsim	≵	\gnsim	∦	\nshortparallel
≨	\lnapprox	≩	\gnapprox	≁	\nsim
⊀	\nprec	⊁	\nsucc	≇	\ncong
⋠	\npreceq	⋡	\nsucceq	⊬	\nvdash
⪵	\precneqq	⪶	\succneqq	⊭	\nvDash
≾	\precnsim	≿	\succnsim	⊮	\nVdash
⪹	\precnapprox	⪺	\succnapprox	⊯	\nVDash
⊊	\subsetneq	⊋	\supsetneq	⋪	\ntriangleleft
⫋	\varsubsetneq	⫌	\varsupsetneq	⋫	\ntriangleright
⫅̸	\nsubseteq	⫆̸	\nsupseteq	⋬	\ntrianglelefteq
⫋	\subsetneqq	⫌	\supsetneqq	⋭	\ntrianglerighteq
↚	\nleftarrow	↛	\nrightarrow	↮	\nleftrightarrow
⇍	\nLeftarrow	⇏	\nRightarrow	⇎	\nLeftrightarrow

표 3.17: AMS 이항 연산 기호

$\dotplus$	\dotplus	$\centerdot$	\centerdot	$\intercal$	\intercal
$\ltimes$	\ltimes	$\rtimes$	\rtimes	$\divideontimes$	\divideontimes
$\Cup$ or $\doublecup$	\Cup or \doublecup	$\Cap$ or $\doublecap$	\Cap or \doublecap	$\smallsetminus$	\smallsetminus
$\veebar$	\veebar	$\barwedge$	\barwedge	$\doublebarwedge$	\doublebarwedge
$\boxplus$	\boxplus	$\boxminus$	\boxminus	$\circleddash$	\circleddash
$\boxtimes$	\boxtimes	$\boxdot$	\boxdot	$\circledcirc$	\circledcirc
$\leftthreetimes$	\leftthreetimes	$\rightthreetimes$	\rightthreetimes	$\circledast$	\circledast
$\curlyvee$	\curlyvee	$\curlywedge$	\curlywedge		

표 3.18: AMS 기타 기호 문자

$\hbar$	\hbar	$\hslash$	\hslash	$\Bbbk$	\Bbbk
$\square$	\square	$\blacksquare$	\blacksquare	$\circledS$	\circledS
$\vartriangle$	\vartriangle	$\blacktriangle$	\blacktriangle	$\complement$	\complement
$\triangledown$	\triangledown	$\blacktriangledown$	\blacktriangledown	$\Game$	\Game
$\lozenge$	\lozenge	$\blacklozenge$	\blacklozenge	$\bigstar$	\bigstar
$\angle$	\angle	$\measuredangle$	\measuredangle	$\sphericalangle$	\sphericalangle
$\diagup$	\diagup	$\diagdown$	\diagdown	$\backprime$	\backprime
$\nexists$	\nexists	$\Finv$	\Finv	$\varnothing$	\varnothing
$\eth$	\eth	$\mho$	\mho		

표 3.19: 수학용 알파벳

예제	명령	포함되어야 할 패키지
ABCdef	\mathrm{ABCdef}	
$ABCdef$	\mathit{ABCdef}	
$ABCdef$	\mathnormal{ABCdef}	
$\mathcal{ABC}$	\mathcal{ABC}	
$\mathscr{ABC}$	\mathcal{ABC}	mathrsfs
$\mathcal{ABC}$	\mathcal{ABC}	eucal 패키지 mathcal 옵션 또는
	\mathscr{ABC}	eucal 패키지 mathscr 옵션
$\mathfrak{ABCdef}$	\mathfrak{ABCdef}	eufrak
$\mathbb{ABC}$	\mathbb{ABC}	amsfonts 또는 amssymb

제4장

특별한 기능

대규모 문서를 LaTeX으로 작성하는 경우, 찾아보기 생성이나 참고문헌 관리와 같은 특별한 기능을 활용하면 많은 도움이 된다. 이와같은 LaTeX의 특별한 기능이나 확장에 대해서는 *LaTeX Manual* [1]과 *The LaTeX Companion* [3]에 보다 자세히 설명되어 있다.

4.1 EPS 그림 포함하기

LaTeX은 `figure`나 `table` 환경으로 떠다니는 표와 그림, 이미지와 그래픽을 다룰 수 있게 해 주는 기본 기능을 제공한다.

기본 LaTeX이나 LaTeX 확장 패키지만으로도 실제 그림을 생성할 수 있는 방법이 몇 가지 있다. 하지만 불행하게도, 대부분의 사용자에게는 너무 난해하다는 게 문제다. 따라서 이 방법은 여기에서 더이상 다루지 않겠다. 이 주제에 관심이 있는 사용자들은 *The LaTeX Companion* [3]이나 *LaTeX Manual* [1]을 참고하기 바란다.

문서 안에 그림을 포함하는 훨씬 쉬운 방법은, 그림 그리기는 전문 그래픽 소프트웨어[1]를 이용하고, 이렇게 해서 완성된 그림을 LaTeX 문서에 삽입하는 것이다. 다양한 그림 삽입 방법을 지원하는 LaTeX 패키지들이 있다. 이 책에서는 그 중 널리 쓰이고 있고 비교적 사용하기 쉬운 Encapsulated PostScript(EPS) 그래픽에 대해서만 다룬다. EPS 형식의 그림을 처리하기 위해서는 포스트스크립트 프린터[2]가 필요하다.

D. P. Carlisle의 graphicx 패키지에는 쓸모있는 명령이 많다. 이것은 "graphics"[3] 패키지 꾸러미(package bundle)의[4] 일부분이다.

---

[1] XFig, CorelDraw!, Freehand, Gnuplot, ... 등.

[2] PS 출력을 얻는 방법으로 GhostScript를 사용하는 방법도 있는데 이 프로그램은 `CTAN:/tex-archive/support/ghostscript`에서 얻을 수 있다. 윈도(Windows) 사용자는 GSview를 이용해서 출력할 수 있을 것이다.

[3] `CTAN:/tex-archive/macros/latex/required/graphics`

[4] LaTeX '패키지'는 `$TEXINPUTS$/latex/required/` 아래 있는 '기본 패키지 꾸러미' (graph-

출력 장치로 PostScript 프린터를 사용하고 graphicx 패키지가 설치되어 있는 경우, 그림을 문서에 포함시키려면 다음과 같은 순서를 차례로 따라하면 된다.

1. 사용하고 있는 그래픽 프로그램에서 그림을 EPS 형식으로 저장한다.[5]

2. 입력 파일의 전처리부(preamble)에서 다음과 같이 graphicx 패키지 사용을 선언한다.

   `\usepackage[`*driver*`]{graphicx}`

   여기서 *driver*는 사용자의 "dvi를 postscript로" 변환하는 프로그램의 명칭이다. 많이 사용되는 프로그램은 `dvips`이다. TEX에는 그림을 붙여넣기 위한 표준이 없으므로 *driver* 이름의 지정이 필요하다. graphicx 패키지는 *driver* 명칭을 알아야 프린터가 처리할 수 있도록 그래픽에 관한 정보를 `.dvi` 파일에 제대로 집어넣어, `.eps` 파일을 정확하게 처리할 수 있게 되는 것이다.

3. 다음의 명령으로 *file*을 사용자 문서에 붙여넣는다.

   `\includegraphics[`*key*=*value*, ⋯`]{`*file*`}`

   옵션 인자 *key*와 여기에 할당하는 변수값인 *value*들은 쉼표(콤마)로 구분하여 입력한다. *key* 값을 지정함으로써 삽입되는 그림의 폭, 높이, 회전값 등을 바꿀 수 있다. 표 4.1에 중요한 *key*를 정리해 두었다.

다음 예제를 살펴보면 좀 더 분명하게 이해할 수 있을 것이다.

```
\begin{figure}
\begin{center}
\includegraphics[angle=90, width=0.5\textwidth]{test}
\end{center}
\end{figure}
```

---

ics, tools, amslatex, psnfss, ⋯)와 `$TEXINPUTS$/latex/contrib/` 아래 놓이는 '추가 패키지군'이 있다. '패키지 꾸러미'라 한 것은 다른 지시가 없으면 '기본 패키지 꾸러미' 가운데 하나를 가리킨다. [역자]

[5]만약 그래픽 소프트웨어가 EPS 저장을 지원하지 않는다면 포스트스크립트 프린터 드라이버를 설치하여 얻는 방법이 있다. 예를 들면 Apple LaserWriter 드라이버를 설치한 다음, 이 드라이버를 이용해서 파일로 인쇄한다. 대개 이렇게 저장한 파일은 EPS 포맷일 것이다. 참고로, EPS 파일은 한 페이지밖에 포함되지 않는다. 프린터 드라이버 설정값을 사용자가 조절하면 EPS 포맷으로 출력하도록 할 수 있는 경우가 많다.

표 4.1: graphicx 패키지의 key 명칭

`width`	그래픽의 폭을 지정
`height`	그래픽의 높이를 지정
`angle`	그래픽을 반시계방향으로 회전
`scale`	그래픽의 배율을 지정

이 코드는 `test.eps`라는 이름으로 저장된 그림을 문서에 삽입한다. 그림은 먼저 90도 각도로 회전시키고, 그 다음에, 그림의 폭이 표준 문단의 0.5배가 되도록 확대하거나 축소한다. 그림의 높이를 특별히 지정하지 않았으므로 가로×세로 비율은 원래 상태가 유지된다. 폭과 높이 값을 각각 절대값으로 지정할 수도 있음은 물론이다. 자세한 내용은 82 쪽에 있는 표 5.5를 참조하기 바란다. 이 문제에 대해 좀더 자세히 알고 싶다면 [8]과 [11]을 살펴보라.

4.2 참고문헌

`thebibliography` 환경을 이용하여 참고문헌 목록을 만들 수 있다. 각 참고문헌 항목은

`\bibitem{`*marker*`}`

으로 시작한다. 문서 본문에서 단행본이나 논문, 기사를 인용하려면 *marker* 라는 명칭으로 참조한다.

`\cite{`*marker*`}`

항목 번호는 자동적으로 붙는다. `\begin{thebibliography}` 명령에 이어 나오는 인자는 자동으로 붙는 번호의 최대 자리수를 지정한다. 아래의 예제에서 `{99}`라고 한 것은 참고문헌 목록의 항목 갯수가 숫자 99의 자릿수(두 자리)를 넘지 않도록 한다는 것을 LaTeX에게 알려주는 것이다.[6]

[6]이 인자로 숫자 이외에 *widest-label*을 지정할 수 있다. *widest-label*은 참조할 레이블이 가질 대강의 길이를 보여주는 텍스트인데, `\bibitem`의 옵션 인자로 주어진 문자열 중에서 제일 긴 것을 쓰면 된다. 이렇게 하면 참고문헌 목록이나 본문의 `\cite`가 쓰인 위치에 숫자가 붙지 않고 이 문자열이 온다.[역자]

```
Partl~\cite{pa} has
proposed that \ldots
\begin{thebibliography}{99}
\bibitem{pa} H.~Partl:
\emph{German \TeX},
TUGboat Volume~9, Issue~1 (1988)
\end{thebibliography}
```

Partl [1] has proposed that ⋯

Bibliography

[1] H. Partl: *German TEX*, TUGboat Volume 9, Issue 1 (1988)

대규모 작업일 경우 BibTEX 사용을 고려해 보는 것도 좋을 것이다. 대부분의 TEX 배포판에는 BibTEX이 포함되어 있다. BibTEX을 이용하면 참고문헌 데이터베이스를 관리할 수 있고, 그 데이터베이스로부터 사용자 문서에서 인용된 것들만 적절히 추출하여 참고문헌으로 제시할 수 있다. BibTEX이 생성하는 참고문헌 목록의 외관(표시방식)은 스타일 양식(style sheets) 개념을 따르고 있기 때문에, 사용자들은 널리 쓰이는 다양한 기존 디자인에 맞는 참고문헌 목록을 작성할 수 있게 된다.

4.3 찾아보기

찾아보기가 붙어 있는 책은 참으로 편리하다. LATEX과 함께 LATEX 지원 프로그램 `makeindex`[7]를 이용하면 매우 손쉽게 찾아보기를 작성할 수 있다. 이 문서에서는 기본적인 찾아보기 목록 생성 명령에 대해서만 설명한다. 보다 상세한 사항은 *The LATEX Companion* [3]을 참고하라.

LATEX의 찾아보기 만들기 기능이 작동하게 하려면, 문서의 전처리부(preamble)에서 makeidx 패키지의 사용을

```
\usepackage{makeidx}
```

와 같이 선언하고, 다음과 같은 특별한 찾아보기 생성 명령어

```
\makeindex
```

를 전처리부(preamble)에 써넣어야 한다.

[7] 파일명에 8자 이상을 사용할 수 없는 시스템에서는 makeidx로 불리기도 한다.

표 4.2: Index Key 사용 예제

예제	찾아보기 항목	설명
`\index{hello}`	hello, 1	기본 항목 생성
`\index{hello!Peter}`	Peter, 3	'hello'의 하위 항목
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	항목 장식
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	위와 같음
`\index{Jenny|textbf}`	Jenny, **3**	페이지 번호 장식
`\index{Joe|textit}`	Joe, *5*	위와 같음

찾아보기의 내용은

`\index{`*key*`}`

로 작성되며, *key*는 찾아보기 항목이다. 최종 찾아보기 목록에서 참조될 본문 위치에서 `\index` 명령을 사용한다. 표 4.2에 몇 가지 예제를 통하여 *key* 인자의 사용법을 설명하였다.

LaTeX은 입력파일을 처리하다가 `\index`명령어가 나타나면 찾아보기 항목과 현재의 페이지 번호를, LaTeX 입력파일과 이름이 같고 확장명이 `.idx`인 특별한 파일에 기록한다. 이 `.idx`화일은 `makeindex` 프로그램을 실행하여 처리한다.

`makeindex` *filename*

`makeindex` 프로그램은 찾아보기를 정렬하여, 파일이름이 같고 확장명이 `.ind`인 파일에 저장한다. LaTeX이 입력파일을 다시 처리할 때, 이렇게 정렬된 색인은 본문내의

`\printindex`

명령이 위치한 곳에 삽입된다.

LaTeX 2_ε의 일부인 showidx 패키지는 찾아보기 항목을 본문의 좌측 여백에 인쇄한다. 이것은 문서의 검토와 찾아보기 확인에 매우 유용하다.

4.4 멋진 쪽머리글(Fancy Headers)

Piet van Oostrum이 작성한 fancyhdr 패키지[8]는 간단한 몇개의 명령어로 문서의 쪽머리글과 쪽바닥글을 사용자가 설정할 수 있게 해 준다. 이 문서의 페이지 상단 쪽머리글이 바로 이 패키지를 활용하여 작성한 것이다.

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
% with this we ensure that the chapter and section
% headings are in lowercase.
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{}  % delete current setting for header and footer
\fancyhead[LE,RO]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\addtolength{\headheight}{0.5pt} % make space for the rule
\fancypagestyle{plain}{%
   \fancyhead{} % get rid of headers on plain pages
   \renewcommand{\headrulewidth}{0pt} % and the line
}
```

그림 4.1: fancyhdr 설정 예제

쪽머리글과 쪽바닥글을 사용자가 조정할 때 실제 장 · 절의 이름이 표시되도록 하고 싶을 때가 있다. 이것은 약간 교묘한 방법을 써야 하는데, LaTeX에서는 2단계 접근 방식으로 이 문제를 해결하고 있다. 머리글과 바닥글을 정의하면서 현재의 장 · 절 이름을 표현하기 위해 `\rightmark`와 `\leftmark`명령어를 사용한다. 이 두 명령어의 값은 chapter나 section명령어가 수행될 때마다 변화하게 된다.

최대한 유연성을 유지하기 위해서, `\chapter` 계열의 명령이 직접 `\rightmark`, `\leftmark`를 재정의하지는 않는다. 그 대신 다른 명령 `\chaptermark`, `\sectionmark` 및 `\subsectionmark`를 호출하는데, 이 명령들이 `\rightmark`와 `\markleft`를 재정의하도록 되어 있다.

따라서, 쪽머리글에서 chapter 이름을 변경하고자 한다면, `\chaptermark`

[8] `CTAN:/tex-archive/macros/latex/contrib/supported/fancyhdr/`에서 구할 수 있다.

를 "새로 지정(renew)"하기만 하면 된다.

fancyhdr 패키지를 써서 이 책의 쪽머리글을 만든 설정 내용을 그림 4.1에 나타내었다. 각주 8에 언급된 주소에서 이 패키지에 관한 문서를 구해볼 것을 권한다.

4.5 Verbatim 패키지

앞에서 verbatim 환경에 대해 알아본 적이 있다. 이 절에서는 verbatim 패키지에 대해 알아보도록 하자. 기본적으로 verbatim 패키지는 원래의 verbatim 환경이 갖는 몇가지 한계를 극복할 수 있도록 verbatim을 재구현한 것이다. 이것 자체는 별로 대단한 것이 아니지만, verbatim 패키지를 구현하면서 몇 가지 기능들이 추가되었는데, 이 추가된 기능 때문에 이 패키지를 언급하는 것이다. verbatim 패키지는 다음 명령, 즉,

`\verbatiminput{`*filename*`}`

을 제공한다. 이 명령은 일반 아스키 텍스트를 마치 verbatim환경에 들어 있는 것처럼 불러올 수 있게 해준다.

verbatim 패키지는 'tools' 패키지 꾸러미에 포함되어 있으므로, 대부분의 시스템에 이미 설치 되어 있을 것이다. 이 패키지에 대하여 좀더 자세히 알고 싶으면 [9]를 참조하라.

4.6 LaTeX 패키지의 다운로드와 설치

대부분의 LaTeX 설치 배포본은 많은 스타일 패키지를 기본적으로 설치해준다. 그러나 인터넷으로 구할 수 있는 수많은 패키지들이 더 있다. 인터넷에서 스타일 패키지를 주로 찾아볼 수 있는 곳은 CTAN(`http://www.ctan.org/`)이다.

geometry, hyphenat 또는 그밖의 여러 패키지들을 대부분 두 개의 파일로 되어 있다. 하나는 `.ins` 확장명을 가진 것이고 다른 하나는 `.dtx` 확장명을 가진 것이다. 보통 `readme.txt`가 있어서 패키지에 대하여 개략적인 설명을 하고 있다. 당연히 이 파일을 먼저 읽어야 한다.

이제 이 파일을 자신의 컴퓨터에 복사한다. 그런 다음에 (a) 자신의 TeX 시스템이 새로운 스타일 패키지를 인식하도록 하고, (b) 관련 문서를 얻는 일련의 절차를 거쳐야 할 일이 남아 있다. 첫번째 작업은 다음과 같이 한다.

1. LaTeX을 `.ins` 파일에 대하여 실행한다. 이렇게 하면 `.sty` 파일이 풀려 나온다.

2. 이 `.sty` 파일을 자신의 TeX 시스템이 스타일 파일을 찾는 위치로 옮긴다. 이 위치는 일반적으로 …/*localtexmf*/tex/latex 디렉토리의 하위

디렉토리이다. (윈도나 OS/2 사용자들은 슬래시를 디렉토리 구분자로 바꾸어서 생각하라.)

3. 파일이름 데이터베이스를 갱신한다. 이 방법은 LaTeX 배포판에 따라 조금씩 다르다. 예를 들면 teTeX, fpTeX의 경우에는 `texhash`이고 web2c에서는 `mktexlsr`이다. MiKTeX은 `initexmf -update-fndb` 명령을 사용하거나 GUI 프로그램을 이용한다.

이제 `.dtx` 파일에서 문서를 추출하여 보자.

1. `.dtx` 파일에 대하여 LaTeX을 실행한다. 이렇게 하면 `.dvi` 파일이 만들어진다. 상호 참조를 정확하게 얻기 위해서는 LaTeX을 몇 번 실행해야 한다는 점에 주의하라.

2. LaTeX이 `.idx` 파일을 컴파일 중에 만들어 내었는지 확인한다. 만약 이 파일이 있으면 아래의 인덱스를 만드는 절차를 거치고 그렇지 않으면 5 단계로 넘어간다.

3. 인덱스를 얻어내기 위해서는 다음과 같이 한다.
   `makeindex -s gind.ist` *name*
   (`name`이 의미하는 것은 주 원본파일이름을 확장명 없이 쓰라는 것이다.)

4. `.dtx` 파일에 대하여 LaTeX을 한 번 더 실행한다.

5. 끝이다. 그렇지만 좀더 문서를 즐겁게 읽으려면 `.ps`나 `.pdf`로 변환하는 과정을 거치는 것이 좋겠다.

어떤 경우에는 `.glo`(glossary) 파일이 만들어질 때도 있다. 이 때는 아래의 명령을 4 단계와 5 단계 사이에서 실행하라.

`makeindex -s gglo.ist -o` *name*`.gls` *name*`.glo`

5 단계까지 가기 전에 마지막 한 번은 항상 `.dtx`에 대하여 LaTeX을 실행해야 한다는 점을 명심하라.

제5장

LaTeX을 자기에게 맞게 바꾸기

앞 장까지 배운 명령으로도, 볼 만한 문서를 만들어낼 수 있다. 멋을 부린 모양은 아니지만, 읽기 쉽고 보기 좋게 만드는 알려진 좋은 조판 규칙을 잘 따르는 문서가 된다.

그런데 LaTeX에서 사용자가 원하는 결과를 얻을 수 있는 명령어나 환경를 제공하지 않는 경우도 있고, 이미 있는 명령을 써서 얻은 출력 결과가 사용자의 요구에 적합하지 않은 경우도 있다.

이 장에서는, LaTeX에서 새로운 값을 정의하는 방법, 그리고 LaTeX 기본값을 이용했을 때 나오는 결과와 좀 다른 모양을 얻고 싶을 때 어떻게 하면 되는지에 대해 약간의 힌트를 제공하려 한다.

5.1 새로운 명령, 환경, 패키지

이 책에서 소개한 명령은 모두 상자 안에 표시하고 책 뒤의 찾아보기에 모아둔 것을 보았을 것이다. 이렇게 하기 위해 LaTeX 필수 명령을 직접 사용하지 않고 새로운 명령과 환경을 정의하여 패키지를 만들었다. 그런 다음 아래와 같이 간단하게 처리하였다.

```
\begin{lscommand}
\ci{dum}
\end{lscommand}
```

\dum

이 예제는 `lscommand`라는 이름의 새로운 환경을 사용하고 있다. 이 새로운 환경은 소개하려는 명령의 이름을 쓰고 그것을 감싸는 상자를 그리는 역할을 한다. 그리고 명령어는 `\ci`라는 새로운 명령으로 지정하였는데, 이것은 명령을 식자하고 동시에 자동으로 찾아보기에 들어가도록 한다. 이 문서 마지막의 찾아보기(index)를 보면 `\dum` 명령어가 나와 있고 `\dum` 명령을 언급한 쪽의 번호가 표시되어 있는 것을 확인할 수 있을 것이다.

만약 명령을 상자 안에 넣는 방식에 싫증이 났다면 lscommand 환경 정의를 새로운 모양이 되도록 살짝 바꾸면 된다. 그러면 이 환경을 사용한 모든 곳의 모양이 다 바뀌는데, 매번 박스를 그리고 명령을 써넣는 식으로 문서를 작성했을 때 그것을 일일이 다 찾아서 바꿔야 하는 것과 비교해보면, 이 방법이 얼마나 편리한 것인지 알 수 있다.

5.1.1 새로운 명령

자신의 새로운 명령을 추가하려면 다음과 같이 한다.

\newcommand{*name*}[*num*]{*definition*}

여기에는 두 개의 인자가 기본적으로 필요하다. 하나는 이 명령의 명칭 *name*이고, 다른 하나는 이 명령의 정의 *definition*이다. 대괄호 안에 있는 *num* 인자는 옵션으로 여러 개의 인자를 가진 명령어를 만들 때 쓰는데, 최대 9개까지의 인자를 가지게 할 수 있다. 만약 이것을 쓰지 않으면 기본값은 0이다. 즉, 인자가 허용되지 않는다.

아래 두 개의 예제는 새로운 명령을 만드는 예를 나타낸 것이다. 첫번째 예제에서는 \tnss라고 하는 새로운 명령을 만들었다. 이것은 "The Not So Short Introduction to LaTeX 2_ε"를 줄여 쓰게 한다. 만약 이 책의 제목을 문서 내에서 자주 사용한다면 이와 같이 줄여서 쓰면 편리하다.

```
\newcommand{\tnss}{The not
    so Short Introduction to
    \LaTeXe}
This is ``\tnss'' \ldots{}
``\tnss''
```

This is "The not so Short Introduction to LaTeX 2_ε" ⋯ "The not so Short Introduction to LaTeX 2_ε"

다음 예제는 *num* 인자의 사용법을 나타낸다. #1은 사용자가 넣는 인자의 내용으로 치환되어 들어가게 된다. 만약 한 개 이상의 인자를 사용하려면, 두번째 인자부터 #2 ⋯ 등으로 쓸 수 있다.

```
\newcommand{\txsit}[1]
 {This is the \emph{#1} Short
      Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
\item \txsit{not so}
\item \txsit{very}
\end{itemize}
```

- This is the *not so* Short Introduction to LaTeX 2_ε
- This is the *very* Short Introduction to LaTeX 2_ε

새로운 명령을 만들었을 때 그 명령이 이미 정의되어 있는 것이라면, LaTeX는 이를 허용하지 않는다. 그러나 \renewcommand 명령을 사용하면 기존에 정의된

내용은 무시되고 새로 정의한 명령이 유효하게 된다. \renewcommand 명령의 사용법은 \newcommand와 동일 하다.

이따금, \providecommand 명령을 써야 할 때도 있다. 이것은 \newcommand와 유사하지만, 만약 명령어가 이미 정의되어 있다면 LaTeX 2$_\varepsilon$는 새로 정의한 것을 무시하고 이미 정의되어 있는 것을 취한다.

LaTeX 명령 뒤에 나오는 공백문자에 대해서 주의할 점이 있다. 7 페이지를 보라.

5.1.2 새로운 환경

\newcommand 명령처럼, 새로운 환경을 만드는 \newenvironment 명령이 있다. 이 명령의 사용법은 다음과 같다.

\newenvironment{*name*}[*num*]{*before*}{*after*}

\newcommand 명령과 마찬가지로, newenvironment에서도 옵션 인자 *num*은 있어도 되고 없어도 된다. *before* 인자에는 이 환경 안에 있는 텍스트가 시작되기 전에 처리할 내용을 넣는다. *after* 인자는 \end{*name*} 명령을 만났을 때 실행될 내용이다.

아래 예제는 \newenvironment 명령의 사용법을 보여준다.

```
\newenvironment{king}
 {\rule{1ex}{1ex}%
      \hspace{\stretch{1}}}
 {\hspace{\stretch{1}}%
      \rule{1ex}{1ex}}

\begin{king}
My humble subjects \ldots
\end{king}
```

■ My humble subjects ⋯ ■

num 인자의 사용법은 \newcommand 명령의 경우와 동일하다. LaTeX은 정의되어 있는 환경이 또다시 정의되는 것을 허용하지 않는데, 만약 이미 있는 환경을 재정의하려면, \renewenvironment 명령을 사용해야 한다. 이 명령의 문법은 \newenvironment 명령과 동일하다.

위의 예제에서 사용된 \rule 명령어는 88 쪽에서 설명하며, \stretch는 81 쪽에서 설명한다. \hspace에 관한 더 자세한 내용은 81 쪽에서 볼 수 있다.

5.1.3 사용자 패키지

만약 새로운 환경과 명령을 많이 정의한다면, 문서의 전처리부가 무척 길어질 수 있다. 이런 경우, 환경과 명령의 정의를 담고 있는 LaTeX 패키지를 만들어 쓰면

좋다. 패키지로 만든 다음에는, \usepackage 명령으로 이 패키지를 문서에 포함하여 사용할 수 있다.

```
% Demo Package by Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short
                       Introduction to \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

그림 5.1: 패키지의 예

패키지를 만드는 것은 기본적으로 문서의 전처리부에 있는 내용을 독립 파일로 복사하는 것이다. 이 파일의 확장명은 .sty가 된다. 다만 한 가지 특별한 명령이 있는데,

\ProvidesPackage{*package name*}

이 명령은 패키지 파일의 가장 앞 부분에서 사용된다. \ProvidesPackage는 패키지의 이름을 LaTeX에 알리는 기능을 함으로써, 컴파일 시 두 번 이상 삽입하려 할 때 이를 알리는 에러 메시지를 내는데 사용된다. 그림 5.1은 위의 예제에서 정의된 명령들을 패키지로 만든것이다.

5.2 글꼴과 크기

5.2.1 글꼴 바꾸기 명령

LaTeX은 문서의 논리적 구조(장/절, 각주 …)에 따라 적절한 글꼴 및 글꼴 크기를 선택한다. 하지만, 사용자가 임의로 글꼴의 모양과 크기를 바꾸고 싶을 경우가 있다. 이에 관한 명령들이 표 5.1과 5.2에 있다. 각 글꼴의 실제 크기는 설계하기 나름이다. 그것은 문서 클래스와 클래스 옵션에 따라 달라진다. 각 표준 문서 클래스에서 이용되는 글꼴 선택 명령의 절대 포인트 크기를 표 5.3에 요약해 두었다.

```
{\small The small and
\textbf{bold} Romans ruled}
{\Large all of great big
\textit{Italy}.}
```

The small and **bold** Romans ruled all of great big *Italy*.

LaTeX 2_ε의 한가지 중요한 특징은 글꼴의 속성이 비의존적이라는 것이다. 달리 말하자면, 이전에 설정한 볼드(굵게)/슬랜트(기울임) 속성을 유지하면서

크기는 물론 글꼴 자체도 바꿀 수 있다는 뜻이다.

수식 모드를 쓰는 경우, 수식 모드를 잠시 빠져 나가 텍스트를 입력하면서 글꼴 바꾸기 명령을 사용할 수도 있다. 수식 조판용 글꼴을 바꾸려면 수식 글꼴 바꾸기를 위하여 특별히 마련된 명령을 써야 한다. 이것은 표 5.4를 참조하라.

글꼴 크기 명령에서 중괄호({)는 중요한 역할을 한다. 이렇게 중괄호로 묶으면 그 안의 내용이 하나의 그룹을 이루게 된다. 대부분의 LaTeX 명령은 이 그룹 안에서 쓰이면 효력이 이 범위 안으로 제한된다.

```
He likes {\LARGE large and
{\small small} letters}.
```

He likes large and small letters.

글꼴 크기 명령은 줄 간격도 바꾼다. 다만 글꼴 크기 명령의 효력 범위 안에 문단 끝(\par 또는 \\)이 올 때만 줄 간격이 바뀐다. 따라서 범위의 끝을 나타내는 중괄호 }가 너무 일찍 나타나면 줄 간격이 바뀌지 않는다. 다음 두 예에서 \par 명령[1]의 위치를 주의깊게 살펴보라.

[1] \par 명령은 빈 줄 하나를 넣는 것과 같다.

표 5.1: 글꼴

\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

표 5.2: 글꼴 크기

\tiny	tiny font	\Large	larger font
\scriptsize	very small font	\LARGE	very large font
\footnotesize	quite small font	\huge	huge
\small	small font	\Huge	largest
\normalsize	normal font		
\large	large font		

표 5.3: 표준 클래스의 글꼴 실제 크기

size	10pt (default)	11pt option	12pt option
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

표 5.4: 수식용 글꼴

Command	*Example*	*Output*
\mathcal{...}	$\mathcal{B}=c$	$\mathcal{B} = c$
\mathrm{...}	K_2	K_2
\mathbf{...}	$\sum x=\mathbf{v}$	$\sum x = \mathbf{v}$
\mathsf{...}	$\mathsf{G\times R}$	$\mathsf{G} \times \mathsf{R}$
\mathtt{...}	$\mathtt{L}(b,c)$	$\mathtt{L}(b, c)$
\mathnormal{...}	$\mathnormal{R_{19}}\neq R_{19}$	$R_{19} \neq R_{19}$
\mathit{...}	$\mathit{ffi}\neq ffi$	$\mathit{ffi} \neq ffi$

```
{\Large Don't read this! It is not
true. You can believe me!\par}
```

Don't read this! It is not true. You can believe me!

```
{\Large This is not true either.
But remember I am a liar.}\par
```

This is not true either. But remember I am a liar.

한 문단 전체나, 그보다 더 긴 텍스트의 일정 부분에 대해 글꼴 크기를 바꾸려 한다면, '환경(environment)' 형태의 구문법을 사용하여 글꼴을 바꾸는 것이 나을 수도 있다.

```
\begin{Large}
This is not true.
But then again, what is these
days \ldots
\end{Large}
```

This is not true. But then again, what is these days …

이렇게 하면 범위를 정해주기 위한 중괄호를 덜 사용해도 된다.

5.2.2 경고, 경고

이 장을 시작할 때 말한 대로, 문서의 각 부분에 이와 같이 구체적이고 개별적인 명령을 어지럽게 삽입하는 것은 위험스런 일이다. 이런 방식은 LaTeX의 기본 개념과 상충하기 때문이다. LaTeX은 문서의 논리적 구성과 그 결과물의 실제 모양을 구별해서 취급한다는 개념을 가지고 있다. 다시 말하면 똑같은 내용의 글꼴 바꾸기 명령이 여러 번 나오는 경우라면 그 명령을 매번 써넣을 것이 아니라 글꼴 바꾸기를 논리적으로 의미하는 `\newcommand`를 정의해서 사용해야 한다는 뜻이다.

```
\newcommand{\oops}[1]{\textbf{#1}}
Do not \oops{enter} this room,
it's occupied by a \oops{machine}
of unknown origin and purpose.
```

Do not **enter** this room, it's occupied by a **machine** of unknown origin and purpose.

이 예제는 위험에 대한 경고문이다. 훗날 위험(물)을 나타내는 데 좀 다른 시각적 표현을 하고 싶을 때, `\textbf`를 일일이 써넣어서 만든 문서라면 그것을 매번 찾아서 확인하고 바꾸어야 할 것이다. 게다가 `\textbf`로 표시된 모든 단어들이 꼭 위험(물)을 나타내는 단어일 것이라는 법이 없으므로 그것들을 일일이 식별해서 바꾸어야 한다. 이에 비하면 `\oops`를 정의해서 한번에 위험물에 대한 표현을 바꾸는 것이 훨씬 쉽고 편리하다.

5.2.3 조언

글꼴과 글자크기에 대한 이야기를 끝내기 전에 조언 한 마디.[2]

> **Remember!** *The* MO RE fonts you use in a document, *the* more READABLE and *beautiful it becomes*.

5.3 간격

5.3.1 줄 간격

만약 문서의 줄 간격을 더 띄우고 싶다면, 다음과 같이 한다.

`\linespread{`*factor*`}`

이 명령은 문서의 전처리부에 넣는다. "한 줄 반"의 줄 간격을 사용하려면 `\linespread{1.3}`으로 하고, "두 줄" 줄 간격으로 벌리려면(영문 원고의 double space) `\linespread{1.6}`을 사용한다. 여기서 *factor* 인자의 기본값은 1이다.

5.3.2 문단의 형식

LaTeX에서 문단 모양(레이아웃)에 영향을 주는 변수가 두 개 있는데. 이 변수의 값을 지정하는 다음 예를 보자.

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

이 두 명령은 입력 파일의 전처리부에 들어간다. 앞의 것은 문단의 첫 줄 들여쓰기 값을 0에 맞추는 것이다.

`plus`와 `minus` 값이 가리키는 것은 TeX이 조판하는 과정에서 문단을 페이지에 정확하게 맞출 수 없을 때 문단 사이의 간격을 줄이거나 늘릴 수 있는 허용범위를 지정해주는 것이다.

유럽에서는 일반적으로 문단과 문단 사이를 넓히고 들여쓰기를 하지 않는다.[3] 이 두 명령은 차례에도 영향을 미쳐서 차례의 줄 사이를 너무 떼어놓는다. 이것을 피하려면, 이 두 명령을 전처리부에 놓지 않고 문서 본문의

[2] "더 많은 글꼴을 사용하면 문서는 더욱 더 읽기 쉽고 아름다워진다." (역자의 사족. 역자는 개인적으로 이 조언이 말하는 바와는 다른 견해를 가지고 있다. 너무 많은 글꼴을 사용한 한글 문서는 오히려 가독성과 품위를 낮추는 결과를 낳을 수도 있는 것이다. 물론 이 문장은 지은이가 그렇게 생각하고 있다는 말이라기보다 글꼴 사용의 예제에 불과하지만.)

[3] 우리나라의 책들은 첫 줄 들여쓰기를 한글 한 글자 정도로 하는 것이 관행인 듯하다. (`\parindent=1em`) LaTeX 표준 클래스들은 영문자 세 글자 정도를 들여쓰는 것을 기본값으로 하고 있다.[역자]

\tableofcontents 다음에 넣거나 아예 사용하지 않을 수 있다. 전문적으로 조판이 잘 된 책을 보면, 첫 줄 들여쓰기를 하고 문단과 문단 사이는 그대로 두는 것이 일반적인 관행이다.

첫 줄 들여쓰기 되지 않은 문단을 강제로 들여쓰기 하려면 다음 명령을 문단 시작 부분에 넣는다.[4]

`\indent`

말할 것도 없이 \parindent의 값은 0 이 아닌 값으로 설정되어야 제대로 동작한다.

첫 줄 들여쓰기를 하지 않으려면 다음 명령을 문단의 시작 부분에 넣는다.

`\noindent`

\section 명령 없이 바로 본문을 시작하고 싶을 때 이 명령을 쓰면 편리하다.

5.3.3 수평 간격

LaTeX은 단어간 간격과 문장 사이 간격을 자동으로 설정한다. 이 간격(여백)을 넓히려면 다음과 같이 한다.

`\hspace{`*length*`}`

\hspace 대신 (별표 붙은) \hspace* 명령어를 사용하면 여백을 넣어야 하는 곳이 줄 끝이나 줄 처음이라도 간격을 유지한다. *length* 인자는 보통 숫자 하나에 단위가 붙은 꼴인데, 주요 단위를 표 5.5에 나타내었다.

```
This\hspace{1.5cm}is a space
of 1.5 cm.
```

This is a space of 1.5 cm.

줄의 끝까지 수평 간격을 넣어 채우는 가변 길이 명령이 있다.

`\stretch{`*n*`}`

두 개의 \hspace{\stretch{*n*}} 명령이 같은 줄에서 사용되면 stretch 인수 n 값에 비례해서 간격이 늘어나게 된다.

```
x\hspace{\stretch{1}}
x\hspace{\stretch{3}}x
```

x x x

[4] 각 장(절)의 첫 문단을 들여쓰기 하려면 indentfirst 패키지를 사용한다. 이 패키지는 'tools' 꾸러미에 포함되어 있다.

표 5.5: TeX의 길이단위

`mm`	millimetre ≈ 1/25 inch
`cm`	centimetre = 10 mm
`in`	inch = 25.4 mm
`pt`	point ≈ 1/72 inch ≈ $\frac{1}{3}$ mm
`em`	사용되고 있는 글꼴에서 'M'자의 폭
`ex`	사용되고 있는 글꼴에서 'x'자의 높이

5.3.4 수직 간격

문단(paragraph), 절(section), 소절(subsection) 등의 수직 간격은 LaTeX에 의해 자동으로 계산된다. 두 문단 사이에 수직 간격을 두어야 할 필요가 있으면 다음 명령을 쓴다.

`\vspace{`*length*`}`

이 명령의 전후에 각각 빈 줄을 하나씩 두는 것이 보통이다. 만약 페이지의 첫 줄 또는 마지막 줄에 걸치더라도 주어진 간격만큼 벌려야 한다면 `\vspace` 대신에 별표 붙은 `\vspace*` 명령을 사용한다.

`\stretch` 명령을 `\pagebreak`와 함께 쓰면 텍스트를 페이지의 마지막 줄에 놓거나 한 페이지의 중앙에 둘 수 있다.

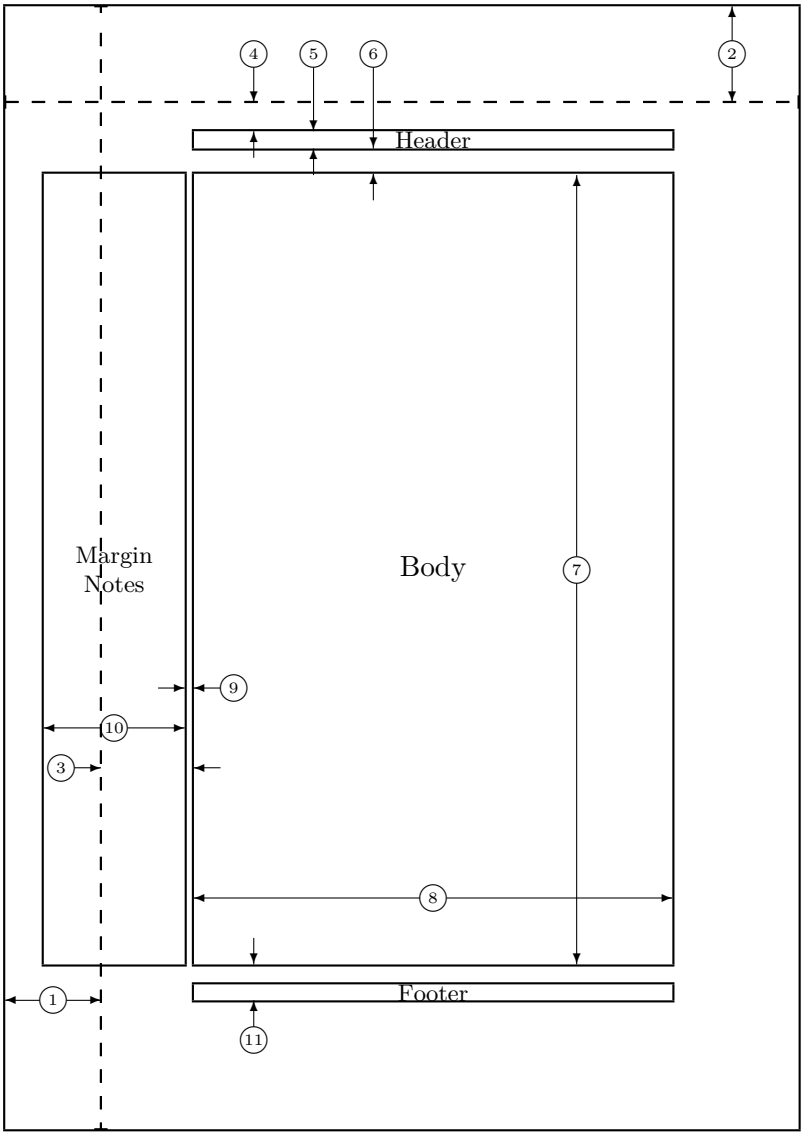
```
Some text \ldots

\vspace{\stretch{1}}
This goes onto the last line of the page.\pagebreak
```

문단을 바꾸지 않고 한 문단 안에서 두 줄 사이를 띄고 싶거나 표에서 두 줄 사이의 간격을 넓히려면 다음 명령

`\\[`*length*`]`

를 사용한다.

`\bigskip`과 `\smallskip` 명령은 정확한 값을 계산하는 번거로움 없이 미리 정의된 크기 만큼 수직 간격을 띄우는 데 쓰인다.



1 one inch + \hoffset
2 one inch + \voffset
3 \evensidemargin = 70pt
4 \topmargin = 22pt
5 \headheight = 13pt
6 \headsep = 19pt
7 \textheight = 595pt
8 \textwidth = 360pt
9 \marginparsep = 7pt
10 \marginparwidth = 106pt
11 \footskip = 27pt
\marginparpush = 5pt (not shown)
\hoffset = 0pt
\voffset = 0pt
\paperwidth = 597pt
\paperheight = 845pt

그림 5.2: 페이지 레이아웃의 변수

5.4 페이지 레이아웃

LaTeX에서는 \documentclass 명령의 인자로 용지 크기를 지정할 수 있다. 그러면 용지의 오른쪽 여백과 본문 영역의 길이를 자동으로 설정한다. 그러나 이렇게 자동 설정된 기정값이 마음에 들지 않는 때도 있을 것이다. 당연히, 이 값을 변경하는 것도 가능하다. 그림 5.2는 변경할 수 있는 변수를 모두 보여준다. 이 그림은 'tools' 꾸러미의 layout 패키지를 써서 만든 것이다.[5]

잠깐! … 이 레이아웃이 너무 좁아 보여서 좀 넓게 만들어야겠다는 분에게 한 마디 하겠다. 잠시 생각해 보자. 다른 LaTeX 기능들이 대부분 그렇듯이, 이처럼 페이지를 좁게 짜는 것이 기본값으로 되어 있는 데는 그럴 만한 이유가 있는 것이다.

MS 워드의 기본 규격 레이아웃과 비교해보면, 확실히 LaTeX의 기본 레이아웃이 상당히 좁다. 그러나 책[6]을 살펴보자. 한 줄에 몇 글자나 들어 있는가? 한 줄에 많아야 66자(영문)를 넘지 않을 것이다. LaTeX 문서를 만들어서 한 줄에 몇 자나 들어가는지 세어보라. 역시 약 66자 정도가 들어가는 것을 알 수 있을 것이다. 실험을 해보면 한 줄에 문자가 많을수록 읽기가 어려워진다는 걸 알 수 있다. 글자가 많을수록 사람의 눈이 줄 끝에서 다음 줄 첫 글자로 이동하기가 어려워지기 때문이다. 신문이 여러 단으로 조판되는 것도 이러한 이유 때문이다.

그러므로, 만약 행의 폭을 늘리면, 독자들이 괴로워진다는 것을 명심하라. 그래도 굳이 문단폭을 넓혀야겠다면, 어떻게 하는 건지 가르쳐주겠다…

LaTeX은 이 변수 값을 바꿀 수 있는 두 개의 명령어를 제공한다. 이 명령들은 문서의 전처리부에서 사용된다.

첫번째는 어떤 고정값을 특정 변수에 할당하는 것이다.

`\setlength{`*parameter*`}{`*length*`}`

두번째 것은 어떤 변수의 기존값에 일정한 값을 더하는 것이다.

`\addtolength{`*parameter*`}{`*length*`}`

이 두번째 명령은 기존값에 대하여 상대 설정이 가능하므로 \setlength 명령어보다 더 유용하다. 예를 들어, 텍스트 영역의 문단폭을 1 센티미터 더 늘리려면 전처리부에 다음과 같이 쓰면 된다.

```
\addtolength{\hoffset}{-0.5cm}
\addtolength{\textwidth}{1cm}
```

이런 일을 하는 데 calc 패키지가 도움이 될 수도 있겠다. 그것은 \setlength의 인수에 대해 산술연산을 할 수 있게 해준다. 길이 설정뿐 아니라 그밖의 다른 함수의 경우에도 인자로서 숫자를 쓸 때는 이 패키지의 도움을 받을 수 있다.

[5] CTAN:/tex-archive/macros/latex/required/tools

[6] 저명한 출판사에서 나온 진짜 인쇄본의 책.

5.5 길이 문제, 몇 가지 더

LaTeX 문서에서는 되도록 절대 길이를 사용하지 않는 것이 좋다. 길이를 지정할 때는 쪽 레이아웃을 구성하는 다른 요소(예를 들면 문단 폭과 같은)의 값을 기초로 해서 상대값을 지정하는 것이 낫다. 그림을 페이지 너비에 맞게 채우고 싶으면 `\textwidth`를 그림의 폭으로 쓰면 될 것이다.

다음 3개의 명령은 어떤 문자열(*text*)의 폭, 높이, 깊이를 계산하여 *command*의 값으로 할당하는 것이다.[7]

```
\settoheight{command}{text}
\settodepth{command}{text}
\settowidth{command}{text}
```

다음 예제는 이 명령을 어떻게 활용할 수 있는지 보여준다.

```
\flushleft
\newenvironment{vardesc}[1]{%
  \settowidth{\parindent}{#1:\ }
  \makebox[0pt][r]{#1:\ }}{}

\begin{displaymath}
a^2+b^2=c^2
\end{displaymath}

\begin{vardesc}{Where}$a$,
$b$ -- are adjunct to the right
angle of a right-angled triangle.

$c$ -- is the hypotenuse of
the triangle and feels lonely.

$d$ -- finally does not show up
here at all. Isn't that puzzling?
\end{vardesc}
```

$$a^2 + b^2 = c^2$$

Where: a, b – are adjunct to the right angle of a right-angled triangle.

c – is the hypotenuse of the triangle and feels lonely.

d – finally does not show up here at all. Isn't that puzzling?

5.6 박스

LaTeX이 조판하는 원리는 페이지에 상자를 하나씩 배치하는 방식이다. 우선 글자 하나하나가 작은 상자이고, 단어는 문자 상자들을 잘 붙여서 만든 상자이다. 이 단어 상자를 다시 다른 단어 상자와 붙여간다. 단어와 단어를 붙일 때는

[7] 폭, 높이, 깊이에 대해서는 94 페이지의 '역자의 보충'을 볼 것.[역자]

약간 특별한 방법을 쓰는데, 이어지는 단어들이 쪽과 줄에 딱 맞아들어가도록 신축성 있게 그 간격을 늘리거나 줄이면서 붙여가는 것이다.

실제로 이루어지는 일은 이보다 더 복잡하지만, 요는 TeX이 상자를 붙여가는 방법으로 식자한다는 것이다. 글자만이 상자인 것은 아니다. 상자에는 무엇이든지 들어갈 수 있다. 심지어 상자 그 자체도 상자에 담을 수 있다. 이렇게 채워진 상자를 LaTeX은 각각 한 글자인 것처럼 취급한다.

상자라고 말하진 않았어도, 앞 장에서 이미 상자를 다루어 본 적이 있다. 예를 들면, `tabular` 환경이나 `\includegraphics` 명령은 상자를 생성한다. 즉, 두 개의 표나 그림을 나란히 붙도록 배열하는 것도 가능하다는 뜻이다. 단지 이렇게 할 때 두 개의 그림이나 표의 폭을 합한 값이 식자폭(textwidth)보다 더 넓어서는 안 된다.

한 문단을 상자 안에 넣으려면 다음 두 가지 방법이 있다. 명령 구문을 이용할 때는 다음과 같이 하고,

`\parbox[`*pos*`]{`*width*`}{`*text*`}`

'환경' 구문을 쓰고 싶으면 다음과 같이 한다.

`\begin{minipage}[`*pos*`]{`*width*`}` text `\end{minipage}`

`pos` 인자로 올 수 있는 것은 상자의 수직 정렬을 조절하는 `c`, `t`나 `b` 문자 중 하나이다. `width` 인자는 상자의 폭을 지정하는 것이다. `minipage`와 `parbox`의 가장 중요한 차이점은, `parbox` 안에서는 사용하지 못하는 명령과 환경이 좀 있지만, `minipage` 안에서는 거의 모든 명령이나 환경을 사용할 수 있다는 점이다.

`\parbox`는 줄바꿈 등등이 포함된 문단 전체를 상자에 넣는다. 수평으로 배열된 것들만을 하나로 묶는 상자 만들기 명령도 있다. 앞에서 본 바 있는 `\mbox`라는 것이 그 가운데 하나인데, 이 명령은 여러 개의 상자를 한 개의 상자로 묶어주기만 할 뿐이다. 그래서 `\mbox`로 묶인 단어들은 줄바꿈이 일어나지 않는다. 상자를 또다른 상자에 넣을 수 있기 때문에, 다음과 같은 수평 상자 묶음 명령을 잘 활용하면 매우 유연하게 써먹을 수 있다.

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

`width` 값은 밖에서 보았을 때 그려지는 상자의 폭이다.[8] 직접 길이값을 써넣지 않고 `\width`, `\height`, `\depth`, `\totalheight`라고 지정할 수도 있다. 이렇게 하면 이 값들은 *text*를 식자한 결과 얻어지는 값을 계산하여 취한다. *pos* 인자로는 `c`, `l`, `r`, `s` 문자 가운데 하나를 취할 수 있는데, 각각 **c**enter(중앙으로),

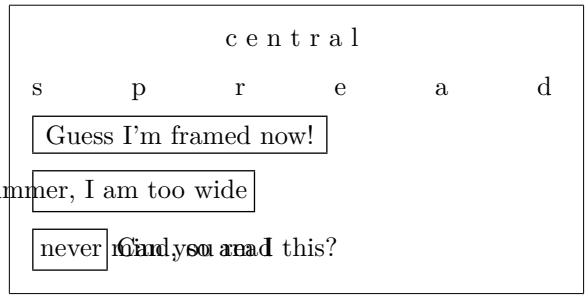
[8] 즉, 이 값이 상자 안의 내용보다 작은 값일 수도 있다. 이 폭을 0pt로 설정할 수도 있는데, 이 때는 박스 안의 텍스트는 바깥쪽 상자와는 관계없이 그냥 출력될 것이다.

left flush(왼쪽으로 몰기), **r**ight flush(오른쪽으로 몰기), **s**pread(크기에 맞추어 펼치기)를 뜻한다.

`\framebox` 명령의 기능은 `\makebox`와 동일하지만, 텍스트 둘레에 상자를 그려준다.

다음 예제는 `\makebox`와 `\framebox` 명령을 활용하여 어떤 일을 할 수 있는지 보여주는 것이다.

```
\makebox[\textwidth]{%
    c e n t r a l}\par
\makebox[\textwidth][s]{%
    s p r e a d}\par
\framebox[1.1\width]{Guess I'm
    framed now!} \par
\framebox[0.8\width][r]{Bummer,
    I am too wide} \par
\framebox[1cm][l]{never
    mind, so am I}
Can you read this?
```



여기까지 수평(horizontal) 위치를 조절하는 방법을 보았다. 다음으로 수직(vertical) 위치를 조절하는 방법을 알아보자.[9] 이것도 LaTeX에서는 어렵지 않다.

`\raisebox{`*lift*`}[`*depth*`][`*height*`]{`*text*`}`

이 명령은 상자의 수직 속성을 정의하게 한다. 첫 세 개의 인자에는 고정값을 넣어도 되지만, `\width`, `\height`, `\depth`, `\totalheight`를 써서 *text*의 크기를 계산하여 그 값으로 동작하게 할 수도 있다.

```
\raisebox{0pt}[0pt][0pt]{\Large%
\textbf{Aaaa\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}%
\raisebox{-4.5ex}{h}}}
he shouted but not even the next
one in line noticed that something
terrible had happened to him.
```

Aaaa$_{\textbf{aa}}$$_{\textbf{r}}$$_{\textbf{g}}$$_{\textbf{h}}$ he shouted but not even the next one in line noticed that something terrible had happened to him.

[9] 수직, 수평 위치를 모두 다루려면 이 각각을 조절하면 된다.

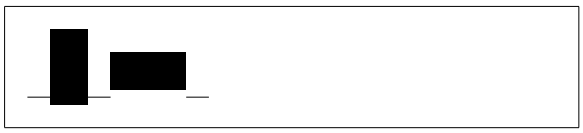
5.7 선 그리기

이 앞 몇 페이지 전에 아래와 같은 명령을 만난 적이 있을 것이다.

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

이 명령은 보통 간단한 검정 상자를 그리는 데 사용한다.

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```



이것은 수직선이나 수평선을 그릴때 유용하다. 예를 들면 이 책 겉표지의 선은 `\rule` 명령을 사용해서 만든 것이다.

높이는 있으나 폭이 없는 특별한 선(`\rule{0pt}{`*height*`}`)이 쓰일 때가 있다. 전문 조판 용어로 strut라는 것이다. 이것은 페이지 상의 어떤 요소가 최소한 일정한 높이를 갖게 하기 위해서 쓰인다. `tabular` 환경에서 행(row) 높이가 적어도 일정값 이상을 갖도록 하는 데 쓸 수 있다.

```
\begin{tabular}{|c|}
\hline
\rule{1pt}{4ex}Pitprop \ldots\\
\hline
\rule{0pt}{4ex}Strut\\
\hline
\end{tabular}
```

참고문헌

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1.

[2] Donald E. Knuth. *The TeXbook,* Volume A of *Computers and Typesetting*, Addison-Wesley, Reading, Massachusetts, second edition, 1984, ISBN 0-201-13448-9.

[3] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8.

[4] LaTeX을 설치할 때는, *LaTeX Local Guide*라는 이름의 문서를 제공하여, 각 로컬 시스템에 특정한 사항들을 설명해야 한다. 그 내용은 `local.tex` 이라는 파일에 들어 있도록 되어 있다. 가끔 게으른 시스템 관리자는 이 문서를 제공하지 않기도 하는데, 이런 경우 주변의 LaTeX 도사(LaTeX guru)에게 물어보는 수밖에 없다.

[5] LaTeX3 Project Team. *LaTeX 2ε for authors.* `usrguide.tex`라는 이름으로 LaTeX 2ε 배포본에 함께 딸려오는 문서이다.

[6] LaTeX3 Project Team. *LaTeX 2ε for Class and Package writers.* `clsguide.tex`라는 이름으로 LaTeX 2ε 배포본에 함께 딸려오는 문서이다.

[7] LaTeX3 Project Team. *LaTeX 2ε Font selection.* `fntguide.tex`라는 이름으로 LaTeX 2ε 배포본에 함께 딸려오는 문서이다.

[8] D. P. Carlisle. *Packages in the 'graphics' bundle.* `grfguide.tex`이라는 이름으로 'graphics' 패키지에 딸려오는 것으로서 LaTeX 배포본을 얻었던 곳에서 찾을 수 있다.

[9] Rainer Schöpf, Bernd Raichle, Chris Rowley. *A New Implementation of LaTeX's verbatim Environments.* `verbatim.dtx`라는 이름으로 'tools'

패키지 꾸러미와 함께 딸려오는 것이다. LaTeX 배포본을 구했던 곳에서 찾을 수 있다.

[10] Graham Williams. *The TeX Catalogue*는 TeX과 LaTeX 관련 패키지 리스트를 거의 전부 망라하고 있다.
CTAN:/help/Catalogue/catalogue.html에서 구할 수 있다.

[11] Keith Reckdahl. *Using EPS Graphics in LaTeX 2$_\varepsilon$ Documents*, 이것은 EPS 그림 파일과 LaTeX에서의 사용에 대해 필요한 내용 대부분과 그 이상의 것을 설명하고 있는 문서이다. CTAN:/info/epslatex.ps에서 구할 수 있다.

역자후기

이 책은 사용법이 쉽지만은 않은 LaTeX의 입문서로 이미 정평이 있다. 비록 양은 얼마 되지 않지만(약 100여 페이지), 논문 작성 등 일반적 용도라면 이 책이 제공하는 정도의 기능만 숙지하더라도 충분히 자신의 목적을 달성할 수 있을 것이다.

이 책을 번역해야겠다는 생각은 오래 전부터 가지고 있었는데, 그것을 실행에 옮길 엄두를 내기가 어려웠다. 우선, 한글판 lshort가 과연 필요할 것인가도 확신하기 어려웠고(왜냐하면 어차피 이 책을 한국어로 옮긴다 하더라도 예제는 여전히 영어 예제를 쓸 수밖에 없으며, 한글 구현에 관한 사항은 이 글의 '번역'에서는 다룰 수 없었기 때문이다.), 사실 초창기 lshort는 영문판도 컴파일이 잘 되지 않는 경우가 있어서, 이것이 과연 한글로 제대로 동작할 것인지 확신할 수 없는 상태였기 때문에, 그냥 영문판을 보는 것으로 만족하고 지낸 것이 사실이다.

그러던 차에, 나의 개인 홈페이지[1] 게시판에서 이 문제를 제기했더니 강윤배 · 장대훈 님이 흔쾌히 돕겠다는 의사를 밝혀 주셨다. 이렇게 의기투합하여, 대부분의 본문을 한글로 옮기는 일을 두 분이 하고, 나는 한글LaTeX으로 컴파일이 되도록 맞추는 일을 주로 하면서 초벌번역이 이루어졌다. 초벌번역이 끝날 무렵, 김재우 님께서는 다른 경로로 나에게 연락을 해오셨는데, 3.01의 번역을 이미 해두신 적이 있다는 것이었다. 이렇게 전체의 번역이 이루어진 후, 내가 각 장을 다시 읽으면서 교열하고 오역을 수정하는 작업을 거쳐 마침내 한국어판 lshort를 출판(!)하게 되었다.

3.07의 번역이 이루어진 뒤, 이 문서가 우리나라의 LaTeX 사용자들에게 상당히 많이 읽혔다는 사실을 알게 되었다. 번역자로서 정말 기쁜 일이다. 우리의 조그만 노력이 LaTeX 사용에 얼마라도 도움을 준 바가 있으면 그것으로 보상을 받은 것이라고 생각한다.

영문판은 3.07 이후에도 계속해서 업데이트가 이루어졌다. 한글판은 그 업데이트를 따라가지 못했고, 그래서 CTAN에 올리는 일도 미루어졌다. 그러던 차에 KTUG(한글 TeX 사용자 그룹) 사이트가 만들어졌고, 이 사이트의 문서화 작업의 일부로 이 문서의 최신판을 한글화하는 일을 다시 시작하게 되었다.

새로운 판의 번역 작업은 3.07의 번역본의 연장선상에서 이루어졌다. 제

[1] `http://www.doeun.pe.kr`

1장은 이재승(berise) 님이, 제2장은 현범석 님이 초벌 번역을 이미 해 둔 것(3.19)이 있어, 그것을 토대로 김강수가 문안을 다듬고 최종판을 만들었다. 다른 장들은 김강수가 대부분 이전의 3.07의 번역문과 같은 것은 조금 손질하는 데 그치고 새로운 부분은 추가로 번역하였다.

이 책이 LaTeX에 입문하는 분들에게 좋은 선물이 되기를 바란다. 사실 한글로 이루어진 TeX 관련서적이 거의 없다 해도 좋을 정도의 상황에서, 이 글이 가치있는 입문서 구실을 충분히 할 것으로 믿는다.

새 판의 번역에서 주의한 것은 다음과 같다.

- 예제들은 영문을 그냥 노출시켰다. 이렇게 한 이유는, 이 책이 LaTeX 2_ε에 대한 설명이지 한글LaTeX에 대한 설명이 아니라는 점 때문이었다. 다시 말하면 이 예제들을 한글화했을 때, 그것은 LaTeX 2_ε를 통해 실행되는 것이 아니라 한글LaTeX을 통해서 실행되는 것이므로, 이 책의 원래 의도와는 동떨어진 것이 된다. LaTeX에서의 한글 사용에 대한 좋은 입문서가 나오기를 바라는 마음 간절하다. 아니 그보다, 안심하고 쓸 수 있는 한글 TeX이 하나 있었으면 하는 생각도 든다.

- 문장의 번역은 무엇보다도 LaTeX 입문자들이 가장 잘 이해할 수 있게 하는 데 초점을 맞추었다. 필요하다면 설명을 길게 덧붙이기도 했고 몇 가지 역자에 의한 보충도 추가하였다. 이런 시도가 도움이 되기를 바란다.

- 용어는 3.07의 번역어를 원칙적으로 그대로 사용했다. 일부 용어들은 이 문서의 한글판에서 처음 번역된 것도 있었고, 그것이 LaTeX 공동체에서 받아들여지고 있는 경우도 있다. 잘못되거나 불일치한 것은 차차 고쳐가겠다.

- 새 판에서는 "한국어 지원" 절(제 2.5.2 절)을 추가하였다. 이 절은 원래 문서의 저자인 Tobias Oetiker씨가 독일어 지원 절(제 2.5.1 절) 대신 자국어 지원으로 바꾸도록 권고한 것을 받아들인 것이지만, 독일어 지원 절 자체를 없애지는 아니하였다. 한국어 지원 절은 김강수가 집필하였다.

- 새 판의 최종 출력물인 PDF는 그 동안 KTUG에서 개발한 TTF 폰트 처리 방식 실험의 성과가 담겨 있고, hyperlink 문제도 해결하여 보기 좋은 온라인 문서를 만들 수 있게 되었다. 이 문제를 해결하기 위하여 원래의 lshort의 스타일 파일을 조금 수정하였다.

책을 옮기는 일은 솔직히 말하면 쉽지 않았다. 한글판 3.07을 번역하는 과정에서 격려해 준 김도현 님, 이현호 님, '무식인' 님을 비롯한 모든 분들에게 특별히 감사의 말을 전한다. 새 판을 내는 데 힘을 보태준 주철 님, 관심을 가져준 이주호 님, 여러 가지 조언을 아끼지 않은 송재훈 님, KTUG의 기둥인 ChoF 님들께도 감사의 말을 전한다. 한글LaTeX의 저자인 은광희 님께 감사한다. 한글LaTeX이 없었으면 이 글의 번역은 불가능했을 것이다.

이 번역본의 모든 책임은 김강수에게 있다. 다른 공동역자들은 오역에 대하여 책임이 없다.

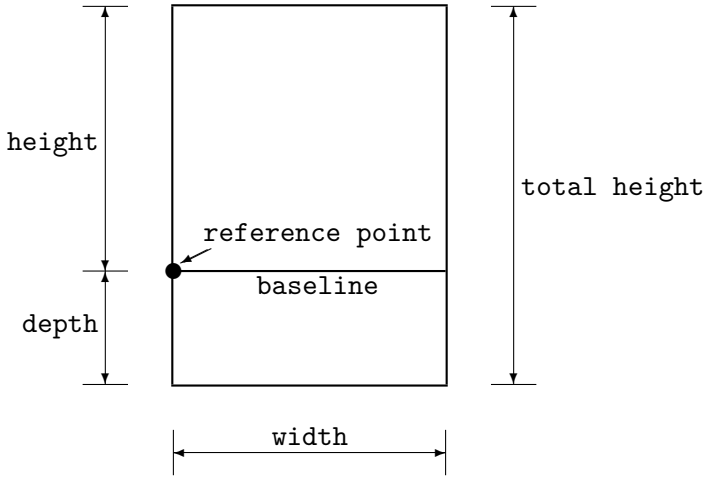
3.07 번역자 : 김강수(karnes@doeun.pe.kr), 강윤배(pear@postech.ac.kr), 장대훈 (webmaster@texworld.pe.kr), 김재우(dbunix@kitinet.co.kr)
3.20 번역자 : 김강수(karnes@doeun.pe.kr), 현범석, 이재승, 주철

공동번역자를 대표하여...
김강수 karnes@doeun.pe.kr

역자의 보충

85 페이지

폭(width), 높이(height), 깊이(depth)는 각각 다음 그림에서 보인 것과 같이 박스의 크기를 나타내는 값들이다.[2]



23 페이지

한글 맞춤법의 '문장부호' 규정에 따르면,

> V. 이음표[連結符]
> 1. 줄표(—)
> 이미 말한 내용을 다른 말로 부연하거나 보충함을 나타낸다.
> (1) 문장 중간에 앞의 내용에 대해 부연하는 말이 끼여들 때 쓴다.
> [예] 그 신동은 네 살에—보통 아이 같으면 천자문도 모를 나이에—벌써 시를 지었다.
> (2) 앞의 말을 정정 또는 변명하는 말이 이어질 때 쓴다.
> [예] 어머님께 말했다가—아니, 말씀드렸다가—꾸중만 들었다.
> 이건 내 것이니까—아니, 내가 처음 발견한 것이니까—절대로 양보할 수 없다.
>
> 2. 붙임표(-)
> (1) 사전, 논문 등에서 합성어를 나타낼 적에, 또는 접사나 어미임을 나타낼 적에 쓴다.
> (2) 외래어와 고유어 또는 한자어가 결합되는 경우에 쓴다.

[2] 김도현 님이 작성한 그림.