The Not So short
Introduction to LaTeX 2ε

LaTeX 2ε 입문

143분 동안 익히는 LaTeX 2ε

by Tobias Oetiker

Hubert Partl, Irene Hyna and Elisabeth Schlegl

Version 6.2, February 28, 2018

김강수, 조인성 옮김

한국어판 2019, 2019년 2월 9일

감사의 말

이 소책자의 많은 부분을 독일어로 쓰여진 오스트리아의 LaTeX 2.09 소개서에서 취하였다. 저자는 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 <noemail>
in Graz

이 독일어 문서에 관심이 있다면 Jörg Knappen이 LaTeX 2_ε를 위하여 업데이트한 버전을 다음 위치에서 찾아볼 수 있다. CTAN://info/lshort/german

아래 열거한 분들은 오류 수정, 내용의 제안, 개선사항의 제공 등을 통하여 이 책자에 기여하였다. 이 책자가 현재의 모습을 갖추는 데 이들이 엄청난 도움을 주었다. 진심으로 감사를 표한다. 당연히 이 책에 있는 모든 오류는 나의 것이다. 제대로 된 곳이 한 군데라도 있다면 그것은 이 분들이 나에게 보내준 내용 중에 들어 있었을 것이 틀림없다.

이 소책자에 기여하고자 한다면 소스코드를 https://github.com/oetiker/lshort 에서 구할 수 있다. 풀 리퀘스트를 환영한다.

Eric Abrahamsen, Lenimar Nunes de Andrade, Eilinger August, Rosemary Bailey, Barbara Beeton, Marc Bevand, Connor Blakey, Salvatore Bonaccorso, Pietro Braione, Friedemann Brauer, Markus Brühwiler, Jan Busa, David Carlisle, Neil Carter, Carl Cerecke, Mike Chapman, Pierre Chardaire, Xingyou Chen, Christopher Chin, Diego Clavadetscher, Wim van Dam, Benjamin Deschwanden Jan Dittberner, Michael John Downes, Matthias Dreier, David Dureisseix, Hans Ehrbar, Elliot, Rockrush Engch, William Faulk, Robin Fairbairns, Johan Falk, Jörg Fischer, Frank Fischli, Daniel Flipo, Frank, Mic Milic Frederickx, David Frey, Erik Frisk, Hans Fugal, Robert Funnell, Greg Gamble, Andy Goth, Cyril Goutte, Kasper B. Graversen, Arlo Griffiths, Alexandre Guimond, Neil Hammond, Christoph Hamburger, Rasmus Borup Hansen, Joseph Hilferty, Daniel Hirsbrunner, Martien Hulsen, Björn Hvittfeldt, Morten Høgholm, Werner Icking, Eric Jacoboni, Jakob, Alan Jeffrey, Martin Jenkins, Byron Jones, David Jones, Johannes-Maria Kaltenbach, Nils Kanning, Andrzej Kawalec, Christian Kern, Alain Kessi, Axel Kielhorn, Sander de Kievit, Kjetil Kjernsmo, Tobias Klauser, Jörg Knappen, Michael Koundouros, Matt Kraai, Tobias Krewer, Flori Lambrechts, Mike Lee, Maik Lehradt, Rémi Letot, Axel Liljencrantz, Jasper Loy, Johan Lundberg, Martin Maechler, Alexander Mai, Claus Malten, Kevin Van Maren, Pablo Markin, I. J. Vera Marún, Hendrik Maryns, Chris McCormack, Aleksandar S. Milosevic, Henrik Mitsch, Stefan M. Moser, Armin Müller, Philipp Nagele, Richard Nagy, Manuel Oetiker, Urs Oswald, Hubert Partl, Marcelo Pasin, Martin Pfister, Lan Thuy Pham, Breno Pietracci, Demerson Andre Polli, Maksym Polyakov, Nikos Pothitos, John Refling, Mike Ressler, Brian Ripley, Kurt Rosenfeld, Bernd Rosenlecher, Chris Rowley, Young U. Ryu, Risto Saarelma, András Salamon, José Carlos Santos, Christopher Sawtell, Gilles Schintgen, Craig Schlenter, Hanspeter Schmid, Baron Schwartz, Jordi Serra i Solanich, Miles Spielberg, Susan Stewart, Matthieu Stigler, Geoffrey Swindale, Laszlo Szathmary, Boris Tobotras, Josef Tkadlec, Scott Veirs, Didier Verna, Carl-Gustav Werner, Fabian Wernli, Matthew Widmann, David Woodhouse, Chris York, Rick Zaccone, Fritz Zaucker, and Mikhail Zotov.

서 문

LaTeX [1]은 과학 및 수학 문서를 작성하는 데 적합한 조판 시스템으로서 대단히 뛰어난 타이포그래피 품질을 얻을 수 있게 한다. 단순한 편지에서 완전한 단행본에 이르기까지 다양한 종류의 문서를 만드는 데도 적합하다. LaTeX은 TeX [2]을 조판 엔진으로 사용한다.

이 길지 않은 입문서는 LaTeX 2_ε에 대해 설명한다. 일반적인 LaTeX 활용에 충분할 정도로 설명할 것이다. LaTeX시스템에 대한 완전한 설명을 보려면 [1, 3]을 참조하라.

이 안내서는 여섯 장(chapter)으로 이루어져 있다.

제 1 장 LaTeX 2_ε의 기본 구조를 설명한다. LaTeX의 역사에 대해서도 조금 알게 될 것이다. 이 장을 읽고 나면 LaTeX이 어떻게 동작하는지에 대해 어렴풋이 이해할 수 있을 것이다.

제 2 장 문서 조판의 세부사항을 다룬다. 필수적인 LaTeX 명령과 환경 거의 대부분을 설명한다. 이 장을 읽고 나면 리스트 문단, 표, 그림, 떠다니는 개체 등을 포함하는 문서를 처음으로 작성할 수 있게 될 것이다.

제 3 장 LaTeX으로 수학식을 식자하는 방법을 설명한다. LaTeX의 가장 강력한 기능인 수식 조판에 대해 다양한 예제를 통해 알려준다. 이 장의 끝에는 LaTeX으로 표현할 수 있는 수학 기호 거의 전부를 표로 정리해 두었다.

제 4 장 색인과 문헌 목록 만들기, 그리고 PDF 생성을 위한 미세설정에 대해서도 약간 다룬다.

제 5 장 LaTeX에서 그림을 그리는 방법을 보여준다. 외부 프로그램으로 그림을 그리고 그것을 파일로 저장하여 문서에 불러들이는 것이 아니라 LaTeX 언어로 그림을 표현하는 방법을 설명한다.

제 6 장 표준 문서 레이아웃을 변경하는, 약간 위험할 수도 있는 내용을 다룬다. LaTeX의 아름다운 출력물을 어떻게 하면 엉망으로 만들거나 (능력에 따라) 더 근사하게 바꿀 수 있는지를 알려준다.

각 장을 순서대로 읽는 것이 중요하다. 이 소책자는 분량이 얼마 되지 않는다. 특히 예제를 주의깊게 보아야 하는데 책자 전체에 걸쳐 나타나는 예제에 많은 중요한 정보가 담겨 있기 때문이다.

LaTeX은 PC, 맥, 대형 유닉스와 VMS 시스템에 이르기까지 거의 모든 컴퓨터에서 이용가능하다. 대학의 컴퓨터실에는 이미 LaTeX이 설치되어 있어서 즉시 사용할 수 있을 것이다. 현재 자신이 이용하는 시스템에 LaTeX이 설치되어 있는지 어떻게 사용하면 되는지 알고 싶으면 *Local Guide* [5]를 보라. 잘 되지 않으면 이 책자를 읽으라고 권한 사람에게 문의하라. 이 책이 다루는 범위는 LaTeX을 통하여 문서를 작성하는 방법에 대해 알려주려는 것이지 LaTeX 시스템을 설치하고 설정하는 문제를 설명하지 않는다.

LaTeX 관련 자료가 필요하다면 Comprehensive TeX Archive Network (CTAN) 사이트를 방문해보라. 홈페이지는 http://www.ctan.org이다.

이 책자 전반에 걸쳐 CTAN에 대한 언급이 나온다. 특히 소프트웨어나 안내문서를 다운로드하도록 지시할 때 그렇다. 완전한 URL을 적는 대신 단지 CTAN:이라고 표시하고 CTAN 트리상의 위치를 표시하였다.

자신의 컴퓨터에서 LaTeX을 실행하고자 한다면 적당한 설치 배포판을 CTAN://systems에서 찾아볼 수 있다. 이 책자의 부록 (A)도 참고하라.

이 문서에 추가하거나 삭제 또는 변경해야 할 부분에 대한 의견이 있으면 저자에게 알려주기 바란다. 특히 이 안내서의 내용이 이해하기 쉬운지 더 좋은 설명 방법은 없을지에 대하여 LaTeX 초보자로부터의 제안을 환영한다.

Tobias Oetiker <tobi@oetiker.ch>

OETIKER+PARTNER AG
Aarweg 15
4600 Olten
Switzerland

이 책자의 최신 버전은 CTAN://info/lshort에서 받아볼 수 있다. 그 하위 폴더 CTAN://info/lshort/korean에서 한국어판을 발견할 수 있을 것이다.

목차

표 목차

그림 목차

제 1 장

알아두어야 할 기본 사항

이 장의 첫 부분에서 LaTeX 2$_\varepsilon$의 철학과 역사에 대하여 간단히 개관한다. 그 다음 부분은 LaTeX 문서의 기본 구조에 초점을 맞추었다. 이 장을 읽고 나면 LaTeX이 동작하는 방식에 대하여 어렴풋이 이해하게 될 것이며 이 책의 나머지 부분도 읽고 이해하고자 하는 필요성을 느끼게 될 것이다.

1.1 간략한 역사

1.1.1 TeX

TeX은 Donald E. Knuth가 만든 컴퓨터 프로그램으로서 텍스트와 수학식을 조판하는 것이 그 목적이다[2]. Knuth는 1977년부터 TeX 조판 엔진을 작성하였는데, 그 무렵 출판 산업에 스며들기 시작하던 디지털 인쇄 장비의 잠재적 가능성을 타진해보려는 의도와, 특히 자신의 책과 논문에서 그가 목도하고 있던 바 타이포그래피적 품질의 저하 경향을 반전시키고자 하는 희망을 가지고 시작한 일이었다. 현재 우리가 사용하는 TeX은 1982년에 발표되었고 1989년에 8비트 문자와 다국어 지원을 위한 약간의 개선을 거쳤다. TeX은 고도로 안정적인 프로그램으로서 여러 종류의 컴퓨터에서 실행되며 거의 버그가 없는 것으로 유명하다. TeX의 버전 번호는 π에 수렴하는데 현재는 3.141592653이다.[1]

TeX은 "테흐(tech)"라고 발음한다.[2] 여기 "ch"는 독일어 "Ach"[3] 또는 스코틀랜드어 "Loch"에서 나는 소리와 같다. "ch"는 그리스어의 "chi"('카이' 또는 '키')라고 하는 글자

[1] [역주] TeX 프로그램의 버전은 3.14159265이고 plain TeX의 버전이 3.141592653이다.

[2] [역주] 우리나라나 영미권에서는 "텍" 또는 "테크"`[tek]`로 발음하는 경우가 많다.

[3] 독일어에는 사실 두 가지 "ch" 발음이 있다. "Pech"의 부드러운 "ch" 소리가 더 적절하지 않으냐고 생각하는 사람도 있다. Knuth에게 이것에 대해 질문하였는데 그는 독일 위키백과에 다음과 같이 썼다. 나는 사람들이 *TeX*을 제각기 좋을 대로 발음한다 해도 화나지 않는다 ... *X*가 모음 *e* 다음에 오기 때문에 독일 사람들은 *a* 다음에 이어지는 거친 *ch* 소리가 아니라 부드러운 *ch*로 소리내는 사람이 많은데. 러시아에서 *'tex'*은 익숙한 단어이고 *'tyekh'*로 발음한다. 그러나 가장 적절한 발음은 그리스어에 있다고 생각하며 거기서는 *ach*와 *Loch*의 거친 *ch* 소리를 들을 수 있다.

χ에서 온 것이다. 또 TEX은 기술(technique)을 뜻하는 그리스 단어의 첫 음절에서 따온 것이다. 아스키로 써야할 적에 TEX을 `TeX`으로 적는다.

1.1.2 LATEX

LATEX은 미리 정의된 전문가 수준의 레이아웃을 사용하여 글쓰는 사람이 자신의 노작을 최고의 타이포그래피 품질로 조판하고 인쇄할 수 있도록 도와준다. LATEX은 원래 Leslie Lamport가 만든 것이었다[1]. LATEX은 TEX을 조판 엔진으로 사용한다. 현재 LATEX은 LATEX 프로젝트 팀이 유지하고 있다.

LATEX은 "레이텍" 또는 "라텍"으로 발음한다. 아스키로 써야할 적에 `LaTeX`으로 적는다. LATEX 2_ε는 "레이텍 투 이"라고 읽으며 `LaTeX2e`로 적는다.

1.2 기초

1.2.1 저자, 북 디자이너, 타입세터

어떤 것을 출판하기 위해서 저자는 출판사에 타자친 원고를 넘겨준다. 그러면 북 디자이너가 책의 레이아웃(문단 폭, 폰트, 표제부 전후의 간격 등)을 결정한다. 북 디자이너는 자신의 지시사항을 원고에 적어넣어서 타입세터(typesetter)에게 넘긴다. 타입세터는 이러한 지시사항에 따라 책을 조판한다.

북 디자이너는 사람인지라 저자가 원고를 쓸 때 어떤 생각으로 쓴 것인지 파악할 수 있다. 자신의 전문적 지식과 원고의 내용에 기초하여 어떤 것이 장의 표제이고 인용이거나 예문인지 혹은 수식인지 등을 판단한다.

LATEX으로 작업하는 환경에서 LATEX은 북 디자이너의 역할을 맡고 TEX이 타입세터가 된다. 그러나 LATEX은 단지 "프로그램일 뿐"이다. 그러므로 가르쳐줘야 할 게 많다. 저자가 자신의 저작의 논리적 구조를 기술하는 추가적인 정보를 제공하여야 한다. 이러한 정보를 "LATEX 명령어"라는 형태로 본문 속에 써넣는다.

이것은 *MS Word*나 *LibreOffice* 등 요즘 대부분의 워드 프로세서가 취하는 WYSIWYG[4]과는 완전히 다른 접근방법이다. 워드 프로세서에서 저자는 컴퓨터에 텍스트를 써넣으면서 동시에 문서 외양(레이아웃)을 눈으로 보면서 조절한다. 인쇄하였을 때 결과물의 모양을 화면으로 보고 있는 것이다.

LATEX으로 작업한다면 내용을 적어넣는 작업 화면이 최종 출력물의 모양으로 보이지는 않는다. 그러나 LATEX으로 파일을 처리한 후에 화면으로 최종 출력물의 모양을 미리보기할 수 있다. 문서를 프린터로 전송하여 실제 출력하기 전에 미리보기를 통해 확인하고 수정하는 것이 가능하다.

1.2.2 레이아웃 디자인

타이포그래피 디자인은 전문분야이다. 비전문가 저자들이 북 디자인이라는 것을 예쁘게만 만들면 되는 거—"예술적으로 멋진 문서가 디자인이 잘 된 것"—라고 생각하기 때문에

[4] What you see is what you get.

심각한 서식 오류를 저지르는 수가 있다. 그러나 책이란 읽혀질 것이지 회랑에 내걸려 있을 것이 아니므로 가독성과 이해가능성은 아름다운 외관보다 훨씬 중요하다. 예를 들면

- 장절 표제의 글자 크기와 번호매김은 독자들이 장절 편제를 통하여 문서의 구조를 명확히 파악할 수 있도록 선택되어야 한다.
- 글줄 길이는 독자의 눈에 부담주지 않을 정도로 짧아야 하지만 판면을 아름답게 보이게 할 정도로 적당히 길어야 한다.

WYSIWYG 시스템은 저자로 하여금 예쁘지만 구조적 일관성이 거의 없는 문서를 생성하게 하는 경향이 있다. LaTeX은 저자가 자신의 문서의 **논리적** 구조를 선언하게 만듦으로써 그런 서식 오류가 발생하지 않게 한다. 가장 적절한 레이아웃을 LaTeX이 선택한다.

1.2.3 장점과 단점

WYSIWYG 세계의 사람들과 LaTeX 사용자가 서로 만나면 "워드 프로세서보다 나은 LaTeX 사용의 장점"이나 그 반대 주제로 토론이 일어나곤 한다. 이런 논란은 대부분 엉뚱한 데로 흐르기 일쑤라 최선의 방책은 가만히 있는 것이지만 때로 피할 수 없는 때도 있는 법…….

그래서 여기 실탄 몇 발을 마련해두려 한다. 일반 워드 프로세서에 비하여 LaTeX이 가진 장점은 다음과 같다.

- 전문가 수준의 레이아웃으로 문서가 마치 실제 출판물처럼 보이게 한다.
- 수학식의 조판과 편리한 입력이 가능하다.
- 사용자는 기억하기 쉬운 문서의 논리 구조를 지시하는 명령 몇 가지를 익히면 된다. 이것으로 문서 레이아웃을 이리저리 끼워맞추는 일을 하지 않을 수 있다.
- 각주, 교차참조, 목차, 문헌목록과 같은 복잡한 구조도 쉽게 생성할 수 있다.
- 기본 LaTeX만으로 수행하기 어려운 타이포그래피적 요구를 충족하는 추가 패키지가 존재하고 자유로이 이용할 수 있다. 예를 들면 PostScript 그래픽을 포함하거나 정확한 출판 표준에 맞는 문헌목록을 조판하도록 하는 패키지가 있다. *The LaTeX Companion* [3]에서 많은 추가 패키지에 대해 설명한다.
- LaTeX은 저자로 하여금 잘 구조화된 텍스트를 쓰도록 유도한다. 구조를 명시하는 것, 그것이야말로 LaTeX 작업의 기본이다.
- LaTeX 2_ε의 조판 엔진인 TeX은 이식성이 뛰어나며 자유 소프트웨어이다. 거의 모든 하드웨어 플랫폼에서도 실행된다.

LaTeX에 단점도 있다. 나로서는 납득할 만한 것이 없지만 수백 가지씩 말하는 사람도 틀림없이 있다. ;-)

- LaTeX은 생각할 머리가 품절인 사람과 잘 맞지 않는다든가…….

- 미리 정의된 문서 서식의 일부 파라미터를 수정하는 것이 가능하기는 한데 새로운 레이아웃을 디자인하는 것은 너무 어렵고 시간이 많이 걸린다.[5]
- 구조적 · 조직적이지 않은 즉흥적 문서를 작성하기가 매우 어렵다.
- 처음에 약간 되는 듯해 보여도 결국 **논리적 마크업**의 개념을 완전히 이해하는 것이 햄스터에게는 무리다.

1.3 LaTeX 입력 파일

LaTeX은 플레인 텍스트 파일로 입력한다. 유닉스나 리눅스 시스템에서 텍스트 파일은 보편적이다. 윈도우즈에서 Notepad(메모장) 응용 프로그램으로 만들고 편집할 수 있다. 입력 파일은 텍스트와 그 텍스트를 조판할 방식을 LaTeX에게 알려주는 명령으로 이루어진다. LaTeX IDE(통합작업환경)은 입력 파일을 텍스트 포맷으로 작성하는 프로그램을 포함한다.

1.3.1 공백

LaTeX은 빈 칸이나 탭 문자 같은 "화이트스페이스" 문자들을 "스페이스" 문자와 완전히 동일하게 다룬다. **여러 개의 이어지는 화이트스페이스 문자는 한 개의 "스페이스"**로 취급한다. 입력 행의 앞부분에 있는 화이트스페이스는 보통 무시된다. 그리고 행 끝의 줄나눔 문자 한 개도 "화이트스페이스"로 본다.

두 행 사이에 빈 줄을 두는 것은 문단의 끝임을 의미한다. **빈 줄 여러 개**를 두어도 **빈 줄 하나**와 동일하게 취급한다. 다음 예제를 보라. 왼쪽에 입력 파일에 입력한 텍스트를 보이고 오른쪽에 그 출력 결과를 나타내었다.

```
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 특별한 문자

다음 기호들은 예약 문자라 부르며 LaTeX에서 특별한 의미로 쓰이거나 폰트로 찍을 수 없는 것이다. 입력 텍스트에 직접 적어넣는다면 대개 인쇄되어 나오지 않을 것이며 LaTeX에게 의도하지 않은 일을 시키는 것이 될 것이다.

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

위에 나온 것처럼 이 글자를 문서에 인쇄되게 하려면 문자 앞에 백슬래시를 붙여서 입력해야 한다.

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

$ % ^ & _ { } ~ \

[5] 소문에 의하면 이 점이 앞으로 나올 LaTeX3 시스템의 주요 요소가 될 것이라 한다.

수학식이나 액센트 붙은 문자 등 특별한 명령으로 입력해야 인쇄되는 부호가 많다. 다른 것과 달리 백슬래시 문자 \는 그 앞에 백슬래시를 하나 더 붙여서(\\) 입력하면 안 된다. 백슬래시 두 개는 강제 줄나눔을 나타내는 명령으로 쓰이기 때문이다. `\textbackslash` 라는 명령을 써야 한다.

1.3.3 LaTeX 명령

LaTeX 명령은 대소문자를 구별한다. 그리고 다음 두 가지 형식 중 하나를 취한다.

- 백슬래시 \로 시작하여 글자(letter)로만 이루어진 이름을 갖는 형태. 명령 이름은 스페이스나 숫자 또는 '글자 아닌 것'이 오면 끝난다.[6]
- 백슬래시 다음에 딱 한 개의 글자 아닌 것으로 이루어진 형태.
- 명령 이름에 별표를 추가하여 "별표 붙은 명령" 형태가 되는 것이 많다.

LaTeX은 명령 뒤의 화이트스페이스를 무시한다. 명령 뒤에 스페이스를 두어야 할 때는 빈 인자 {}를 붙이거나 특별한 스페이스 명령을 사용해야 한다. 빈 인자 {}는 LaTeX이 명령 이름 직후의 스페이스를 잡아먹지 못하도록 만든다.

```
New \TeX users may miss whitespaces
after a command. % renders wrong
Experienced \TeX{} users are
\TeX perts, and know how to use
whitespaces. % renders correct
```

New TEXusers may miss whitespaces after a command. Experienced TEX users are TEXperts, and know how to use whitespaces.

인자(parameters)를 요구하는 명령이 있다. 인자는 명령 이름 뒤에 중괄호 { }로 묶어서 전달한다. 옵션 인자(optional parameters)를 취하는 경우도 있는데 이것은 명령 이름 뒤에 대괄호 [ ]에 묶어서 전달한다. 인자는 보통 명령 자체가 요구하는 것이므로 생략할 수 없지만 옵션 인자(선택적 인자)는 생략가능하다.

command[*optional parameter*]{*parameter*}

LaTeX 명령의 사용례를 다음 보기에서 보였다. 지금 무슨 명령인지 모르겠다고 걱정할 필요 없다. 나중에 다 설명한다.

```
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!

[6] [역주] 여기서 말하는 '글자'와 '글자 아닌 것'에 대하여 부언한다. TEX은 입력되는 문자(토큰)를 몇 개의 범주(category)로 구분하여 처리한다. 그 가운데 "letter"라고 부르는 범주가 있다. 즉 TEX 매크로 명령의 이름은 "letter" 범주에 속하는 문자로만 이루어진다는 의미이다. "letter" 범주에 속하는 문자의 범위는 TEX 엔진에 따라 달라지는데 전통적·표준적으로 오직 영문자 알파벳(아스키 문자)만이 속한다고 생각하면 된다. 숫자나 기호문자는 "글자"의 범주에 들지 않는다. 이 번역본에서 "글자"라는 말이 TEX category를 의미할 적에는 "**글자**(*letter*)"와 같이 표시했다.

1.3.4 주석(Comments)

LaTeX이 입력 파일을 처리하던 중에 % 문자를 만나면 그 줄의 나머지 부분과 줄나눔 문자 그리고 다음 줄 시작 부분의 화이트스페이스를 무시한다.

이를 이용하여 입력 파일에 주석이나 메모를 적어두는 데 사용할 수 있다. 이 부분은 출력되지 않는다.

한 문장을 여러 줄로 나누어 입력할 때 나누어지는 위치에 있는 개행 문자나 화이트스페이스를 무시하도록 하는 데 % 문자를 이용할 수도 있다.

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

This is an example: Supercalifragilisticexpialidocious

verbatim 패키지가 제공하는 comment 환경을 이용하여 더 긴 주석문을 작성할 수도 있다. \usepackage{verbatim}이라는 문장을 문서의 전처리부에 적고 다음 예제와 같이 이 명령을 사용한다.

```
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.

이것은 예컨대 수학식같은 좀더 복잡한 환경 안에서는 동작하지 않을 수 있음을 알아두자.

1.4 입력 파일의 구조

LaTeX 2ε는 입력 파일을 제대로 처리하기 위해 그것이 일정한 구조를 갖추고 있을 것을 요구한다. 그래서 입력 파일의 맨 처음에 다음 문장이 있어야 한다.

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

이 문장은 작성하는 문서가 어떤 종류의 것인지를 지정하는 것이다. 이 다음에 전체 문서의 형식에 영향을 주는 명령이나 LaTeX에 새로운 기능을 추가하는 패키지를 로드하는 문장이 온다. 패키지를 로드하려면

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

라고 쓴다.

설정 작업이 다 되면[7] 이제 문서의 본문을 시작한다는 뜻으로 다음 명령을 준다.

```
\begin{document}
```

[7] \documentclass와 \begin{document} 사이의 영역을 **전처리부**(*preamble*)라고 한다. 문서 전체에 대한 설정 작업이 이루어지는 곳이 전처리부이다.

이 다음에 문서의 내용을 텍스트와 적당한 LaTeX 명령을 함께 섞어서 작성한다. 문서의 끝에는

```
\end{document}
```

라는 명령을 두어야 하는데 이것은 LaTeX에게 작업의 끝임을 알려주는 역할을 한다. 이 명령 이후에 오는 내용은 어떤 것이든 다 무시된다.

그림 1.1이 가장 간단한 LaTeX 2_ε 입력 파일을 보여주고 있다. 그림 1.2는 조금 더 복잡한 보기이다.

한국어 독자를 위하여 한국어-한글 문서의 최소 작성례를 그림 1.3에 보인다.

1.5 명령행 작업

8페이지에 보인 간단한 LaTeX 입력 파일을 처리하는 방법이 무척 궁금할 것이다. 그 방법을 알아보자. LaTeX 자체는 뭔가 누를 수 있는 멋진 버튼을 갖춘 GUI 프로그램이 아니다. 입력 파일을 처리하는 것말고는 보여주는 것이 없다. 일부 LaTeX 작업환경을 제공하는 프론트엔드 프로그램에 컴파일 버튼이 달린 GUI가 제공되기도 하지만 명령행 인터페이스만을 갖춘 시스템도 있다. 여기서는 텍스트 기반 시스템에서 LaTeX을 불러서 입력 파일을 처리하도록 하는 방법에 대해서 살펴보겠다. 다만, 아래 기술은 LaTeX이 컴퓨터에 이미 설치되어서 잘 작동하고 있는 경우를 가정한다.[8]

1. LaTeX 입력 파일을 만들고 편집한다. 반드시 플레인 텍스트 파일이어야 한다. 유닉스 시스템의 모든 편집기가 플레인 텍스트 파일을 만든다. 윈도우즈에서는 **플레인 텍스트** 형식으로 저장하도록 해야 한다. 파일 이름을 선택하고 확장명으로 `.tex`을 부여한다.[9]

2. 셸 또는 cmd 창을 연다. `cd` 명령으로 저장된 파일이 있는 디렉터리(폴더)로 찾아 들어가서 이 파일에 대하여 LaTeX 명령을 실행한다. 실행하여야 하는 LaTeX 명령은 `xelatex`이나 `lualatex` 또는 `pdflatex`이다.[10] 성공적으로 실행이 이루어지면 파일이름이 같은 `.pdf` 파일을 얻을 수 있다. 목차나 교차참조 등을 처리하기 위해서 LaTeX을 두 번 이상 실행해야 할 수도 있다. 만약 입력 파일에 오류가 있으면 LaTeX 프로그램이 이를 알려주면서 처리를 중단할 것이다. 이럴 때는 `ctrl-D`를 눌러 명령행으로 돌아간다.[11]

   ```
   xelatex foo.tex
   ```

[8] 잘 관리되고 있는 유닉스 시스템에서라면 당연히 그러할 것이다. 그리고 참된 인간이라면 모름지기 유닉스를 사용하는 법…… ;-)

[9] [역주] 한국어 사용자를 위한 첨언. 한국어 윈도우즈 사용자는 파일이 UTF-8 인코딩으로 저장되도록 주의를 기울여야 한다. Notepad 앱을 사용한다면 저장시에 이를 선택할 수 있다. 레이텍 전용 편집기는 UTF-8로 저장되는 것이 기본이지만 편집기의 설정에서 UTF-8 저장이 활성화되어 있는지를 확인하는 것이 좋다. "ANSI 인코딩"으로 저장하면 안 된다. 리눅스나 유닉스는 시스템의 기본 언어가 UTF-8이라면 크게 신경쓰지 않아도 된다. 그리고 파일 이름은 영문 아스키 문자만으로 짓는 것이 좋다. 한글 이름은 피하도록 하라.

[10] [역주] 2019년 현재 한국어 문서 작성을 위하여 권장되는 것은 `xelatex`이다.

[11] [역주] 오류는 이를 수정하여 다시 시도하여야 한다. 많은 오류가 사소한 오타에서 비롯되므로 이를 찾아 수정하면 된다. 오류의 종류와 그 해결 방법에 대해서는 이 책을 끝까지 읽으면 더 많이 알 수 있게 된다.

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

그림 1.1: LaTeX 파일의 최소 작성례

```
\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{Some Interesting Words}
Well, and here begins my lovely article.
\section{Good Bye World}
\ldots{} and here it ends.
\end{document}
```

그림 1.2: 실제 저널 논문의 예. 이 예제에 나온 명령은 이 책의 나중에 모두 설명한다.

```
\documentclass[a4paper]{article}
\usepackage{kotex}
\author{저자명}
\title{최소 작성례}
\begin{document}
\maketitle
\tableofcontents
\section{서론}
우리는 라텍을 배우기 시작했다. 이제 첫 문서를
작성한다.
\section{결론}
일찌감치 끝낸다.
\end{document}
```

그림 1.3: 한글 문서의 최소 작성례. UTF-8 유니코드 인코딩으로 저장하여야 한다.

1.6 문서 레이아웃

1.6.1 문서 클래스

LaTeX이 요구하는 첫 번째 정보는 작성자가 만들려고 하는 문서의 유형에 대한 것이다. `\documentclass` 명령으로 이를 지정하여 준다.

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

여기서 *class*라는 것은 만들어질 문서의 종류를 나타내는 것이다. 표 1.1에 기본적인 문서 클래스를 열거하였다. LaTeX 표준 배포판에는 이밖에 letter와 slide라는 클래스가 더 있으며 다른 목적의 문서 클래스도 많다. *option* 파라미터는 문서 클래스의 동작 방식을 사용자가 지정하는 값들이다. 옵션은 쉼표로 분리하여 열거해야 한다. 표준 문서 클래스에 적용할 수 있는 흔히 쓰이는 옵션을 표 1.2에 보였다.

지금까지 설명을 바탕으로 예를 들어 보자.

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

이 명령이 지시하는 바는 "`article` 클래스"의 문서를 작성하되, 기본 폰트 크기는 "11 포인트"로 하고 "펼침면(양면) 조판" 형식으로 "A4 용지"에 맞추라는 것이다.

1.6.2 패키지

문서를 작성하다보면 기본 LaTeX만으로는 해결하기 어려운 문제를 만날 수 있다. 예컨대 그래픽을 포함해야 한다든가 색깔있는 텍스트를 쓴다든가 소스 코드를 파일로부터 읽어서 문서에 넣는다든가 하는 경우에 기본 LaTeX의 기능을 확장해야 할 필요가 생긴다. 이러한 기능 확장은 패키지를 통하여 이루어진다.

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

패키지를 활성화하는 명령은 이와 같다. 여기서 *package*는 패키지의 이름이고 *options*는

표 1.1: 문서 클래스

`article`	과학 학술지 논문, 발표자료, 짧은 보고서, 프로그램 문서, 초대장 등을 위한 클래스
`proc`	article 클래스에 기반한 프로시딩용 클래스
`minimal`	가장 기본적인 클래스. 페이지 크기와 기본 폰트만을 설정한다. 오류추적 등을 위해 주로 사용한다.
`report`	장(chapter)을 포함하는 클래스. 긴 보고서, 소책자, 박사논문 등에 쓸 수 있다.
`book`	단행본 제작을 위한 클래스
`slides`	슬라이드용 클래스. 산세리프체 큰 글씨를 기본 글꼴로 한다. 실제 슬라이드 제작에는 이것보다 beamer 클래스를 더 많이 쓴다.

그 패키지의 특정 기능과 동작을 제어하기 위한 키워드의 목록이다. `\usepackage` 명령은 전처리부에서만 쓸 수 있다. 1.4절을 참고하라.

LaTeX 2_ε 기본 배포판에 포함되어 딸려오는 패키지가 몇 가지 있다. (표 1.3을 보라.) 별도로 제공되는 것도 많다. 어떤 패키지가 자신의 시스템에 설치되어 있는지는 *Local Guide* [5]를 참고할 수 있다.[12] LaTeX 패키지에 관한 주요한 정보는 *The LaTeX Companion* [3]에서 얻을 수 있는데 수백 가지 패키지와 그것이 하는 일에 대하여 설명하고 있다.

요즘 TeX 배포판은 엄청난 수의 패키지를 한꺼번에 설치해준다. 각 패키지 문서를 참조하려면 `texdoc` 명령을 사용한다.

표 1.2: 문서 클래스 옵션

`10pt`, `11pt`, `12pt` 문서 본문 폰트 크기를 설정한다. 옵션을 따로 주지 않으면 `10pt`.

`a4paper`, `letterpaper`, ... 용지 크기를 설정한다. 기본값은 `letterpaper`이다. 이 밖에 `a5paper`, `b5paper`, `executivepaper`, `legalpaper`를 줄 수 있다.

`fleqn` 별행 수식을 왼쪽 정렬로 식자한다. 이 옵션을 주지 않으면 가운데 정렬한다.

`leqno` 수식에 붙는 번호를 왼쪽에 붙인다. 이 옵션을 주지 않으면 오른쪽에 수식 번호가 인쇄된다.

`titlepage`, `notitlepage` 문서 표지를 별도의 한 페이지로 만들고 내용을 새 페이지로 시작할 것인지 그러지 않을 것인지를 지정한다. `article` 클래스는 표지면을 별도로 만들지 않는 것이 기본값이며 `report`와 `book`은 별도 페이지로 하는 것이 기본이다.

`onecolumn`, `twocolumn` 1단이나 2단 조판을 선택한다.

`twoside`, `oneside` 단면문서인지 펼침면 조판[double sided]인지를 지정한다. `article` 클래스와 `report`는 단면이 기본이고 `book`은 펼침면 조판이 기본이다. 이 옵션이 의미하는 바는 문서의 모양을 어떻게 만들 것이냐에 관한 것일 뿐이고 프린터에게 양면인쇄를 하라는 명령을 보내는 것은 아니라는 점을 알아두자.

`landscape` 가로가 긴 페이지 레이아웃(landscape)을 선택한다.

`openright`, `openany` 장(chapter)이 오른쪽 페이지(홀수면)에서 시작하게 할 것인지 홀짝수면의 구분 없이 시작할 수 있게 할 것인지 선택한다. `openright`를 선택하면 이전 장의 마지막이 홀수면일 때 그 다음에 짝수면 하나를 내용없이 채우고 새 chapter를 다음 홀수면에서 시작하게 된다. 이것은 `article`에서는 동작하지 않는 옵션이다. `report`는 다음 페이지에서 바로 새로운 장을 시작하는 것이 기본값이고 `book`은 항상 오른쪽 페이지에서 시작하는 것이 기본값이다.

[12] [역주] 일괄 설치 배포판 TeX Live에 어떤 패키지가 있는지 알아보려면 https://ctan.org/pkg에서 검색해볼 수 있다.

표 1.3: LaTeX 기본 배포 패키지

doc	LaTeX 프로그램의 문서화를 가능하게 하는 패키지이다. `doc.dtx`[a]와 *The LaTeX Companion* [3]에 상세 설명이 있다.
exscale	수학 기본 폰트의 스케일 버전을 제공한다. `ltexscale.dtx`에 설명이 있다.
fontenc	폰트 인코딩을 지정한다. `ltoutenc.dtx` 문서에 설명이 있다.
ifthen	'if...then ...otherwise ...' 형식의 명령을 지원한다. `ifthen.dtx`와 *The LaTeX Companion* [3]을 보라.
latexsym	LaTeX 기호 폰트를 사용가능하게 한다. `latexsym.dtx`와 *The LaTeX Companion* [3]을 보라.
makeidx	색인 작성을 위한 패키지이다. 이 책자의 4.2절과 *The LaTeX Companion* [3]에 설명되어 있다.
syntonly	LaTeX이 문법 검사만 하고 실제 출력물을 생성하지 않게 한다.
inputenc	입력 인코딩을 설정하게 한다.[b] `inputenc.dtx`에 설명이 있다.

[a] 이 파일은 시스템에 당연히 설치되어 있으며 `latex doc.dtx`를 실행하면 쓰기 권한 있는 폴더 어디서나 `dvi` 파일을 얻을 수 있다. 다른 패키지에 대해서도 마찬가지이다.

[b] [역주] 앞서 역자주에서 언급한 대로 오늘날 라텍 시스템은 UTF-8을 읽고 쓸 수 있기 때문에 입력 인코딩은 중요한 문제가 아니게 되었다. 그러나 여전히 레거시 텍의 활용 빈도가 높은 상황에서는 이 패키지가 중요한 역할을 한다.

1.6.3 페이지 스타일

LaTeX은 세 종류의 상하단 면주 형식을 제공한다. 면주 형식을 페이지 스타일이라고 부른다.

`\pagestyle{`*style*`}`

여기서 *style* 위치에 올 수 있는 인자는 `plain`, `headings`, `empty` 가운데 하나이다. 표 1.4에 미리 정의된 페이지 스타일을 열거하였다.

표 1.4: LaTeX의 페이지 스타일

`plain`	페이지 하단부 중앙에 페이지 번호를 인쇄한다. 기본 페이지 스타일이다.
`headings`	각 페이지 상단에 페이지 번호와 장 표제를 인쇄하고 하단부는 비운다. (이 책자에서 사용하고 있는 스타일과 비슷하다.)
`empty`	페이지의 상단과 하단을 모두 비우고 면주에 아무 것도 없게 한다.

현재 페이지의 페이지 스타일을 바꾸려면

`\thispagestyle{`*style*`}`

이라고 명령한다. 스스로 페이지의 상하단 면주를 설계하여 새로 만들 수도 있는데 그 방법을 *The LaTeX Companion* [3]과 이 책자 69페이지 4.3절에서 찾을 수 있다.

1.7 파일과 확장명

LaTeX 작업을 하다보면 마주치게 되는 문제로 여러 가지 확장명을 가진 파일들이 생겨나는데 그게 무엇이고 왜 만들어졌는지 알 수 없다는 점이 있다. 다음에 보이는 목록은 TeX 작업 중에 볼 수 있는 파일 종류를 설명한다. 이 목록에 있는 것이 모든 확장명 파일을 다 설명하고 있지는 않다. 중요한 확장명 파일에 대한 설명이 빠져 있다고 생각하면 저자에게 알려주기 바란다.

.tex LaTeX 또는 TeX 입력 소스 파일. TeX 문서의 기본 확장명이다.

.sty LaTeX 매크로 패키지. 이 파일은 `\usepackage` 명령으로 문서 중에 로드할 수 있다.

.dtx TeX 문서화 파일. LaTeX 스타일 파일은 이 형식으로 배포한다. .dtx 파일을 컴파일하면 스타일 파일에 관련된 문서를 얻을 수 있다.

.ins .dtx 파일에 포함되어 있는 파일을 풀어내기 위하여 필요한 지침을 적은 인스톨 보조 파일. 온라인으로부터 LaTeX 패키지를 내려받았을 때 보통 .dtx와 .ins로 이루어져 있다. .ins에 LaTeX을 적용하면 .dtx로부터 필요한 파일을 풀어낸다.

.cls 클래스 파일. `\documentclass` 명령으로 지정할 수 있다.

.fd 폰트 기술 파일.

다음은 LaTeX을 실행할 때 작업 폴더에 생성되는 파일들이다.

.dvi '장치 독립(Device Independent)'에서 온 확장명으로서 전통적인 LaTeX 컴파일 결과 생성되는 출력 파일이다. DVI 프리뷰어 프로그램으로 화면상으로 결과를 보거나 `dvips`와 같은 유틸리티 프로그램을 이용하여 프린터로 보내거나 한다. 그러나 pdfLaTeX이나 최근의 새로운 엔진을 적용한 LaTeX을 주로 쓰는 요즘은 이 파일을 보기가 힘들어졌다.

.log 컴파일 과정에서 일어난 상세한 기록을 담은 로그 파일.

.toc 장절 표제를 저장하는 파일. 다음 번 컴파일 때에 목차를 생성하기 위해 이 파일을 읽는다.

.lof .toc와 같은 종류의 것이며 그림의 목록을 담고 있다.

.lot 마찬가지로 표의 목록을 담고 있다.

.aux 컴파일할 적에 다음 번 컴파일을 위하여 전달해야 할 정보를 적어두는 파일. 특히 교차참조에 필요한 정보를 저장하고 있다.

.idx 색인을 만들고 있다면 LaTeX은 모든 색인용 단어들을 이 파일에 저장한다. 이 파일을 `makeindex` 프로그램으로 처리하여 색인을 만든다. 색인 만들기와 관련해서 68페이지의 4.2절을 참고하라.

.ind .idx 파일을 처리하여 문서에 들어갈 색인을 담고 있는 파일. 다음 번 컴파일 때 포함된다.

.ilg `makeindex`를 실행한 로그 파일.

1.8 큰 규모의 글쓰기

큰 문서를 작업할 때에 여러 부분으로 입력 파일을 나누어놓는 것이 좋다. LaTeX은 이런 작업에 필요한 두 가지 명령을 제공한다.

`\include{`*filename*`}`

이 명령을 본문에 쓰면 *filename.tex*이라는 이름의 외부 파일의 내용을 그 위치에 삽입한다. *filename.tex*에 포함된 내용을 처리하기 직전에 페이지가 나누어진다는 사실을 기억하라.

다음 명령은 전처리부에서 쓸 수 있다. 이 명령으로 지정된 파일에 대한 `\include` 명령이 본문에 나올 경우에 그것만을 포함하라는 의미이다.

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

전처리부에 이 명령이 놓여 있으면 여기에 열거된 파일에 대한 `\include` 명령만이 실행된다. 다른 파일을 불러들이는 `\include` 명령이 더 있다고 하더라도 무시된다.

`\include` 명령은 외부 파일을 새로운 페이지를 열어서 포함시킨다. `\includeonly`를 쓰는 경우에, 해당 파일이 존재하지 않는 경우라 하더라도 페이지가 나누어지는 위치가 변하지 않기 때문에 유용하다. 그러나 가끔 페이지나눔 없이 파일을 포함하여야 할 때가 있다. 이럴 때는 다음 명령을 사용한다.

`\input{`*filename*`}`

명령이 주어진 위치에 다른 조치 없이 해당 파일을 바로 포함한다.

syntonly 패키지를 이용하여 문서를 빠르게 검토할 수 있다. 문서에서 명령의 사용법이 올바른지 구문 오류는 없는지 체크하기만 하고 출력물 (pdf) 파일을 만들지 않는다. 이 검토는 매우 빠르게 이루어지므로 귀중한 시간을 절약할 수 있게 해주는데 특히 터미널로 문서를 작성하는 경우에 유용하다.

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

출력물을 얻으려면 두 번째 줄을 주석처리(줄 앞에 퍼센트 기호 %를 추가)하면 된다.

제 2 장

텍스트의 조판

앞 장을 읽고서 LaTeX 문서의 기초 사항을 알게 되었다. 이 장에서는 실제 제대로 된 문서를 작성하기 위해 알아두어야 할 문서 구조에 대하여 다룬다.

2.1 텍스트와 언어의 구조

글쓴이: Hapspeter Schmid <hanspi@schmid-werren.ch>

글쓰기의 주목적은 저자의 사상, 정보, 지식을 전달하는 것이다. 내용이 잘 구조화되어 있을수록 독자가 이해하기 쉽다. 또한 타이포그래피적 요소가 내용의 논리적 의미적 구조를 잘 반영하고 있을수록 그 구조를 바로 파악하기 쉬워진다.

LaTeX은 텍스트의 논리적 의미론적 구조만을 지시한다는 점에서 다른 조판 시스템과 다르다. LaTeX은 문서 클래스 파일과 여러 스타일 파일에서 주어지는 "규칙"에 따라 텍스트의 조판 형태를 만들어낸다.

LaTeX에서 (일반적으로 타이포그래피에서) 가장 중요한 텍스트 단위는 문단이다. 이것을 "텍스트 단위"라 하는데 그 이유는 문단이 한 가지 일관된 생각이나 개념을 반영하는 타이포그래피상의 형태이기 때문이다. 이어지는 소절에서 예컨대 \\를 이용하여 줄을 바꾸거나 또는 빈 줄을 두어 문단을 구분하는 방법을 배우게 될 것이다. 그러므로 새로운 문단이 시작되는 것은 새로운 생각이 시작되었을 때여야 한다. 그런 것이 아니면 문단 구분이 아니라 줄나눔만을 써야 한다.[1] 문단을 구분하여야 할지 어떨지 잘 모르겠다면 텍스트가 개념이나 사고의 흐름을 전달하는 매개체라는 관점에서 살펴보라. 문단이 나누어졌는데 이전의 생각이 이어지고 있다면 그 문단나눔을 제거해야 한다. 완전히 새로운 사유를 전개하는데 문단이 나누어지지 않았다면 거기서 문단을 나누어야 한다.

[1] [역주] 'line breaking'의 역어로서 이 번역본에서는 "줄나눔"과 "행나눔"을 혼용하였다. '개행(改行)'이라 하기도 한다.

많은 사람들이 문단나눔을 적절히 하는 것이 얼마나 중요한지를 잘 모르고 있다. 문단나눔의 의미조차 알지 못하는 사람도 많고 특히 LaTeX 사용자 중에 그것이 문단나눔인지조차 알지 못하고 문단을 나누는 사람도 있다. 텍스트 안에 수식이 사용될 때 특히 이런 실수를 자주 저지른다. 다음 예를 보고 수식을 전후하여 어떨 때 빈 줄(문단나눔)이 들어가고 어떨 때 들어가지 않았는지 이해하자. (여기 사용된 명령 중에 모르는 것이 있다면 이 장과 다음 장을 읽은 후에 되돌아와서 다시 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 Kirchhoff'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
```

문단보다 작은 텍스트 단위는 문장이다. 영문에서 약어 뒤의 마침표보다 문장의 종지를 의미하는 마침표에 더 큰 공백을 둔다. LaTeX은 입력된 마침표가 어떤 경우에 해당하는지를 결정하려 하는데 LaTeX의 선택이 적절하지 않으면 글쓴이가 원하는 바를 지시해주어야 한다. 이 장의 뒷부분에서 설명한다.

문장은 더 작은 요소들로 구분된다. 대부분의 언어에서 문장부호 사용법은 매우 복잡하다. 그렇지만 영어나 독일어 같은 많은 언어에서 쉼표 사용법은 그것이 언어의 흐름상 구분점을 의미한다는 것을 기억하면 거의 실수없이 적용할 수 있다. 쉼표를 어디에 찍어야 할지 모르겠다면 문장을 큰 소리로 읽으면서 쉼표가 있는 위치에서 짧게 숨을 쉬어보자. 그 숨쉬기가 어색하게 느껴진다면 그곳의 쉼표는 지운다. 만약 어딘가 숨을 쉬거나 잠시 휴지(休止)를 두어야겠다고 느낀다면 거기에 쉼표를 삽입한다.

마지막으로 문단은 더 상위 수준인 장(chapter), 절(section), 소절(subsection) 등을 구성하도록 논리적 구조에 따라 나열되어야 한다. 예를 들어 `\section{The Structure`

of Text and Language}라고 적어 넣는 것이 무엇을 의미하는지 그리고 이 어구가 어떻게 배치될지 명령 자체가 이미 다 설명하고 있다.

2.2 줄나눔과 페이지나눔

2.2.1 문단 정렬

각 행의 길이가 같도록 양끝맞춤(justified)으로 정렬하는 것이 보통 문서의 조판 관행이다. 이를 위하여 LaTeX은 전체 문단의 내용을 최적화하여 단어 사이에 줄나눔과 공백을 삽입한다. 필요하다면 한 줄에 잘 들어맞지 않는 단어를 하이픈처리한다. 문단을 조판하는 모양은 문서 클래스를 따른다. 문단의 첫 줄을 들여쓰기하고 문단과 문단 사이에 추가 간격을 두지 않는 것이 일반적이다. 103페이지의 6.3.2절에서 자세히 다룬다.

특별한 경우에 LaTeX에게 강제로 행을 나누도록 해야 할 때가 있다.

`\\` 또는 `\newline`

이 명령은 새로운 문단을 시작하지 않은 상태에서 줄나눔을 행한다.

다음 명령은 강제로 줄나눔을 행하되 페이지나눔은 일어나지 않도록 하는 것이다.

`\\*`

새 페이지를 시작하려면 다음 명령을 쓴다.

`\newpage`

줄나눔과 페이지나눔에 관련된 명령 몇 가지가 있다.

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

선택인자 n은 0에서 4까지의 값을 가질 수 있는데 이 값을 이용하여 명령이 영향을 미치는 강도를 조절할 수 있다. n을 4 미만으로 설정하는 것은 LaTeX이 조판 결과가 좋지 않으면 이 명령을 무시해도 좋다는 의미이다. “break”와 “new” 명령을 혼동하면 안된다. “break” 명령이 입력되었을 때 LaTeX은 여전히 줄의 오른쪽 끝과 페이지의 아래쪽 끝을 가지런하게 정렬하려고 시도하기 때문에 좋지 않은 조판 결과를 얻게 될 수 있다. 이에 대해 다음 절에서 설명한다. 정말로 “새로운 줄”이나 “새로운 페이지”를 시작하려 한다면 “new” 명령을 써야한다. 이름이 그렇게 붙여진 이유가 있는 것이다.

LaTeX은 가능한 한 최선의 줄나눔을 하려고 한다. 만약 LaTeX이 설정한 높은 기준을 충족하는 줄나눔이 실패하면 문단에서 그 한 줄이 오른쪽으로 튀어나가도록 조판한다. 그러면서 “overfull hbox”라는 경고 메시지를 컴파일 과정에 보여준다. LaTeX이 단어의 적절한 분철(hyphenate) 위치를 찾지 못했을 때 자주 일어난다.[2] LaTeX에게 그 기준을 좀

[2] LaTeX이 `Overfull \hbox`가 발생한 줄 번호와 경고메시지를 보여주기는 하지만 해당 위치를 찾는 것이 항상 쉬운 것은 아니다. `\documentclass` 명령의 옵션으로 `draft`를 지정하면 이런 일이 발생한 줄의 오른쪽 여백에 두꺼운 검은 선을 그려서 표시해준다.

낮추라고 하려면 \sloppy 명령을 준다. 이렇게 하면 튀어나가는 줄은 사라지지만 결과가 적절하지 않더라도 단어 사이의 간격을 늘려서 조판하면서 "underfull hbox" 경고를 보여주므로 대부분의 경우에 출력되는 결과는 그다지 좋지 않다. LaTeX이 원래의 기준으로 조판을 행하도록 되돌리려면 \fussy 명령을 준다.

2.2.2 분철

LaTeX은 필요할 때 단어를 분철(hyphenate)한다. 분철 알고리즘이 적절한 분철 위치를 찾지 못하면 TeX에게 예외 처리를 지시하는 다음 명령을 써서 문제를 해결한다.

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

이 명령의 인자로 열거되는 단어들은 "-"로 표시된 위치에서만 분철이 이루어진다. 인자로 오는 것은 LaTeX이 일반 문자(letter)로 간주하는 문자와 기호를 포함해야 한다. 분철 위치는 hyphenation 명령이 주어진 위치에서 활성화된 언어 용으로 저장된다. 즉 hyphenation 명령을 문서의 전처리부에 두었다면 영어 언어 전체의 분철에 영향을 미친다. 이 명령을 \begin{document} 이후에 두고 polyglossia 같은 다국어 지원 패키지를 활용한다면 polyglossia에 의하여 활성화된 언어에서만 동작하게 할 수 있다.

다음 예는 "hyphenation"과 "Hyphenation"에 동일한 분철 규칙을 적용하고 "FORTRAN", "Fortran", "fortran"에 대해서 분철을 허용하지 않게 한다. 특수문자나 기호는 허용되지 않는다.[3]

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

\- 명령은 단어에 임의의 분철 규칙을 삽입한다. 그리고 그 위치가 해당 단어에서 유일하게 분철 가능한 위치가 된다. 이 명령은 특수문자(예: 강세표시 문자)를 포함하고 있는 단어에서 특히 유용한데 특수문자를 포함하는 단어에 대하여 LaTeX이 자동으로 분철하지 않기 때문이다.[4]

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

I think this word is: supercalifragilisticexpialidocious

몇 단어를 묶어서 줄나눔을 하지 않도록 하려면 다음과 같이 한다.

`\mbox{`*text*`}`

인자로 주어진 단어들은 항상 묶인 상태가 된다.

[3] [역주] 각 단어 사이에 , 등의 구분 기호가 없음에 주의.

[4] [역주] 강세표시 문자가 포함된 단어를 분철하지 못한다는 것은 OT1 인코딩의 폰트를 사용하는 경우에 해당한다. 현대의 유니코드 텍엔진이나 T1 인코딩 폰트를 사용하는 경우에 이런 문자가 포함된 단어에 대해서도 분철이 가능하다. 한편 한국어 표기는 분철부호를 쓰지 않으며 모든 음절문자 사이에서(일부 예외를 제외하면) 줄나눔을 할 수 있다.

```
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.

\fbox는 \mbox와 비슷한데 인자로 주어진 단어에 테두리를 그려준다.

2.3 미리 정의된 문자열

앞서 몇몇 예제에서 특정 문자열을 식자하는 LaTeX 명령을 이미 보였다. TeX, LaTeX의 로고를 식자하는 명령이 포함되어 있다.

명령어	출력 예	설명
\today	February 13, 2019	시스템의 오늘 날짜[5]
\TeX	TeX	최고의 조판 시스템
\LaTeX	LaTeX	지금 배우고 있는 것의 이름
\LaTeXe	LaTeX 2$_\varepsilon$	LaTeX의 현재 버전

2.4 특수문자와 기호

2.4.1 따옴표

따옴표를 입력하기 위해 타자기에서처럼 "를 사용해서는 **안 된다**. 출판물에서 사용하는 여는 따옴표와 닫는 따옴표는 모양이 다르다. LaTeX은 두 개의 `(grave accent)로 여는 따옴표를, 두 개의 '(vertical quote)로 닫는 따옴표를 표시한다. 작은따옴표는 이것을 한 번씩 사용한다.

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

“Please press the ‘x’ key.”

편집기 상에서 (폰트에 따라 다르겠지만) 여는 따옴표가 back-tick이나 grave accent(`)이고 닫는 따옴표가 quote(')인 것이 마음에 들지 않을 수도 있지만 그렇게 입력해야 하는 것으로 기억하자.[6]

2.4.2 대시와 하이픈

LaTeX에 네 가지 대시가 있다. 세 가지는 잇대어 입력하는 대시의 개수에 따라 달라진다. 네 번째 것은 사실 대시가 아니라 수학 부호이다.

[5] [역주] 한국어 TeX에서 \today 명령이 만들어내는 문자열은 “2019년 2월 12일”과 같은 모양일 수 있다. 이 날짜 문자열은 polyglossia 패키지가 해당 언어에 따라 생성할 수도 있다.

[6] [역주] TeXShop이나 TeXworks 같은 LaTeX 전용 편집기는 사용자의 따옴표 입력 편의를 위해 키보드의 "를 연달아 입력해도 지능적으로 `` ''로 바꾸어주는 기능이 있다. XeLaTeX을 위해 유니코드 따옴표 “ ”로 바꾸어주기도 한다. 그러나 소스에 키보드 따옴표 문자 "가 그대로 입력되어서는 안 된다는 것은 유효하다.

```
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

각각 '-' 하이픈, '–' 엔대시, '—' 엠대시, − 뺄셈 부호라고 부른다.[7]

2.4.3 틸데 (~)

웹 주소에서 흔히 보이는 문자 ~가 틸데이다. LaTeX에서 `\~{}`로 입력하면 ~를 출력해주지만 원하던 것이 아닐 수 있다. 다음 예를 보라.[8]

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

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

2.4.4 슬래시 (/)

두 단어 사이에 슬래시를 넣으려면 간단히 `read/write`와 같이 입력할 수 있다. 그렇지만 이렇게 하면 이 전체를 하나의 단어로 인식하게 된다. 단어 내부에서 분철이 일어나지 않기 때문에 'overfull' 경고를 만날 수 있다. `\slash` 명령을 사용하여 예컨대 `read\slash write`와 같이 입력하면 분철이 허용되게 된다. 그러나 '/' 문자는 분수나 단위를 나타낼 때는 그대로 사용해야 한다. `5 MB/s`.

2.4.5 도 기호 (○)

도 기호를 입력하는 방법을 다음 예에서 볼 수 있다.

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

It's −30 °C. I will soon start to super-conduct.

또는, `\textdegree` 명령으로 입력하는 것도 가능하다. `\textcelsius` 명령은 도 기호 뒤에 C를 붙여준다.

```
30 \textcelsius{} is
86 \textdegree{}F.
```

30 ℃ is 86 °F.

이 경우에, 만약 XeLaTeX이나 LuaLaTeX이 아닌 legacy TeX(pdfLaTeX)으로 컴파일하고 있다면 textcomp 패키지를 usepackage 하여야 한다.

[7] [역주] 한글 맞춤법에 이 부호와 유사한 모양의 '줄표'와 '붙임표'가 있다. 길이에 따른 대시의 종류 구분은 없으며 사용법에도 차이가 있다. 예를 들어 영어 문장에서는 범위를 나타내기 위하여 엔대시를 쓰지만 한글 맞춤법에서는 이 자리에 '물결표(~)'를 쓰도록 규정하고 있다. 붙임표는 분철(하이프네이션)과는 아무 관련 없고 이어지는 내용을 묶어 열거할 때 사용하는 것이다. 엠대시를 우리 글의 줄표(어구를 나누거나 강조 생략하기 위해 쓰는 부호)를 쓸 자리에 쓸 수 있다.

[8] [역주] 웹 주소를 문서에 적어넣을 적에 틸데 문자를 그대로 사용할 수 있게 하는 url 또는 hyperref 패키지의 `\url` 명령을 쓰는 것이 좋다.

2.4.6 유로 통화 기호(€)

이제 금융 관련한 글을 쓰려면 유로화 기호가 필요한 시대가 되었다. 오늘날 많은 폰트들이 이 기호를 이미 포함하고 있으므로 다음과 같이 하는 것으로 충분하다.

```
\texteuro
```

€

레거시 텍 엔진(pdfLaTeX)을 쓰고 있다면 textcomp 패키지가 필요하다.

이 번역본의 대본이 된 영문판 lshort에는 유로화 기호에 대한 다양한 표현 방법을 소개하고 있으나 일부는 XeLaTeX과 같은 현대적 TeX 엔진에서 불필요하거나 호환되지 않는 것이고 우리 실정에 꼭 들어맞는다고 할 수 없어서 번역본에서는 해당 부분을 제외하였다.

참고로, 원화 기호와 엔화 기호는 다음과 같이 표현할 수 있다. (마찬가지로 textcomp 패키지를 요구할 수 있다.)

```
\textwon, \textyen
```

₩, ¥

2.4.7 줄임표 (. . .)

타자기에서는 쉼표나 마침표가 다른 글자와 같은 폭을 차지하지만 인쇄된 서적에서는 이 글자들이 그 앞 글자에 아주 가깝게 붙는다. 그러므로 '줄임표'를 나타내기 위하여 점 세 개로 표현하면 제대로 표현되지 않는다. 이를 위한 별도의 명령이 있다.[9]

`\ldots` (low dots)

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

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

2.4.8 합자

라틴문자의 인쇄에 있어 낱글자를 각각 식자하지 않고 몇 글자를 묶어서 하나의 활자로 식자하는 경우가 있다. 이것을 합자(ligature)라고 한다. 실제로는 특수기호를 사용하여 식자하는 것이다.

ff fi fl ffi. . . instead of ff fi fl ffi . . .

합자로 식자하지 않고 각 글자를 낱낱이 찍게 하려면 `\mbox{}`를 합쳐지는 글자 사이에 넣는 방법이 있다. 두 단어가 합쳐져 이루어진 단어의 경계 위치에서 필요할 수 있다.

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

Not shelfful
but shelfful

[9] [역주] 한글 맞춤법에서 규정하고 있는 '줄임표'는 여섯 개의 가운뎃점으로 표현한다. 이를 위해서 kotex 패키지가 재정의한 `\ldots`를 두 번 써야 한다……·.

2.4.9 액센트와 특수 문자

LaTeX은 여러 언어의 액센트와 특수 문자를 지원한다. 표 2.1은 o 문자에 다양한 액센트를 붙인 예를 보여준다. 다른 글자에도 마찬가지로 적용 가능하다.[10]

i와 j 위에 액센트를 두려 할 때 점을 제거해야 할 필요가 있다. \i, \j와 같이 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: 액센트와 특수 문자

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	o	\c o
o̩	\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 다국어 지원

글쓴이: Axel Kielhorn <A.Kielhorn@web.de>

LaTeX으로 영어가 아닌 다른 언어로 문서를 작성하는 데 있어 다음 세 가지가 마련되어 있어야 한다.

1. 자동 생성되는 문자열[11]이 해당 언어에 알맞게 적용되어야 한다.
2. 해당 언어의 분철 규칙을 LaTeX이 알 수 있도록 해야 한다.
3. 언어와 문자마다 나름의 타이포그래피 규칙이 있다. 예컨대 프랑스어에서는 각 콜론(:) 앞에 공백을 꼭 두는 것이 관행이다.

또한 자신의 언어로 된 텍스트를 입력함에 있어서 표 2.1에서 보인 명령들을 일일이 타이핑하는 것은 좀 귀찮은 일이다. 이런 문제를 극복하기 위해서 최근까지 입력 인코딩이며 폰트 인코딩이라는 골치아픈 영역을 알아야 했다. 그러나 오늘날 현대적 TeX 엔진은 자연스럽게 UTF-8을 읽고 쓸 수 있게 되었으며 문제점들은 상당한 정도로 완화되었다.

[10] [역주] XeTeX이나 LuaTeX과 같은 유니코드 텍 엔진을 사용할 적에 액센트 붙은 문자는 유니코드 문자를 그대로 입력해도 잘 처리한다. ò ó ô õ ö ø.

[11] 목차(Table of Contents), 그림 목차(List of Figures), ...

```
\usepackage{iftex}
\ifXeTeX
   \usepackage{fontspec}
\else
   \usepackage{luatextra}
\fi
\defaultfontfeatures{Ligatures=TeX}
\usepackage{polyglossia}
```

그림 2.1: LuaLaTeX과 XeLaTeX을 위한 전처리부 다국어 처리 일괄 설정

polyglossia[18]는 babel 패키지를 대체하는 패키지로서 각 언어의 분철 패턴과 자동 생성 문자열을 처리해준다.

fontspec[20]은 XeLaTeX과 LuaTeX에서 폰트 사용을 제어한다. 기본 폰트는 Latin Modern Roman이다.

2.5.1 Polyglossia 사용법

다중언어를 적절히 처리하게 하기 위해 문서의 전처리부에 적어야 하는 명령이 TeX 엔진에 따라 조금씩 다르다. 22페이지의 그림 2.1은 전처리부에서의 언어 설정에 관한 예시이다.

그 동안은 유니코드 TeX 엔진을 사용해서 얻을 이득이 별로 없었다. 이 상황은 라틴 문자를 벗어나서 그리스어나 러시아어와 같은 흥미로운 언어를 만나면서 바뀌게 되었다. 유니코드에 기반을 둔 시스템을 사용하면 에디터에 고유 문자를 손쉽게[12] 입력해넣을 수 있다. 그리고 TeX이 그것을 이해한다.

다국어로 글쓰는 것은 간단하다. 전처리부에 해당 언어를 특정하기만 하면 된다. 다음 예에서 csquotes 패키지를 사용한 것을 볼 수 있는데 사용 중인 언어에 따라 적절한 유형의 인용부호를 생성해준다. 이 패키지를 언어 설정보다 **앞에** 두어야 한다는 점을 기억하자.

```
\usepackage[autostyle=true]{csquotes}
\setdefaultlanguage{english}
\setotherlanguage{german}
```

이제 독일어 문단은 german 환경으로 쓸 수 있다.

```
English text.
\begin{german}
Deutscher \enquote{Text}.
\end{german}
More English \enquote{text}.
```

English text. Deutscher „Text“. More English “text”.

다른 언어로 한 단어 정도를 적으려면 \text*language* 명령을 쓸 수 있다.

[12] 손쉽다는 말의 최소한도의 의미에서

```
Did you know that
\textgerman{Gesundheit} is
actually a German word.
```

Did you know that Gesundheit is actually a German word.

이 예의 경우는 별 필요없는 일로 보일 수 있지만 그래도 적절한 분철을 얻는다는 장점이 한 가지 있다. 그러나 제2의 언어가 약간 특이한 것이라면 노력을 들일 필요가 없지 않을 것이다.

문서에서 사용된 폰트에 제2 언어에서 요구하는 글리프가 빠져 있을 수가 있다. Latin Modern 폰트를 예로 들자면 키릴 문자를 포함하고 있지 않다. 해결책은 해당 언어에 적합한 폰트를 하나 정의하는 일이다. polyglossia 패키지는 새로운 언어가 활성화되면 먼저 그 언어 용으로 정의된 폰트가 있는지를 체크한다. computer modern 폰트가 마음에 든다면 다음 내용을 문서 전처리부에 두어서 "Computer Modern Unicode" 폰트를 사용하게 할 수 있다.

LuaLaTeX은 간단하다.

```
\setmainfont{CMU Serif}
\setsansfont{CMU Sans Serif}
\setmonofont{CMU Typewriter Text}
```

XeLaTeX을 위해서는 조금 더 구체적으로 설정한다.

```
\setmainfont{cmun}[
   Extension=.otf,UprightFont=*rm,ItalicFont=*ti,
   BoldFont=*bx,BoldItalicFont=*bi,
 ]
 \setsansfont{cmun}[
   Extension=.otf,UprightFont=*ss,ItalicFont=*si,
   BoldFont=*sx,BoldItalicFont=*so,
 ]
 \setmonofont{cmun}[
   Extension=.otf,UprightFont=*btl,ItalicFont=*bto,
   BoldFont=*tb,BoldItalicFont=*tx,
 ]
```

폰트 설정이 적절하게 이루어지면 다음과 같이 할 수 있다.

```
\textrussian{Правда} is
a russian newspaper.
\textgreek{ἀλήθεια} is truth
or disclosure in philosophy
```

Правда is a russian newspaper. ἀλήθεια is truth or disclosure in philosophy

xgreek 패키지[21]는 고전 그리스어와 현대 그리스어(monotonic 또는 polytonic) 처리를 지원한다.

오른쪽에서 왼쪽으로 쓰는(RTL) 언어

왼쪽에서 오른쪽으로 쓰는 것이 보통이지만 오른쪽에서 왼쪽으로 쓰는(RTL) 언어가 있다. RTL 언어를 식자하기 위하여 polyglossia 패키지는 bidi 패키지를 요구한다. (bidi는

LuaTEX을 지원하지 않는다.) bidi 패키지는 보통 마지막에 로드하는 hyperref보다도 뒤, 마지막에 위치하여야 한다. (polyglossia가 bidi를 로드하기 때문에 polyglossia도 마지막으로 로드하는 패키지가 되어야 한다.)

xepersian[23] 패키지는 페르시아어를 지원한다. `\section`과 같은 LATEX 명령을 페르시아어로 입력할 수 있도록 하기 때문에 원어 사용자에게 매력적이다. 이 패키지는 XELATEX을 통하여 카시다[13]를 식자할 수 있는 유일한 패키지이다. 비슷한 알고리즘을 사용하는 시리아어 패키지가 개발 중에 있다.

SCICT[14]이 제공하는 IranNastaliq 폰트는 웹사이트 http://www.scict.ir/Postal/Home/Default.aspx에서 내려받을 수 있다.

arabxetex[19] 패키지는 아랍 문자를 사용하는 여러 언어를 지원한다.

- arab (Arabic)
- persian
- urdu
- sindhi
- pashto
- ottoman (turk)
- kurdish
- kashmiri
- malay (jawi)
- uighur

이 패키지는 ArabTEX 아스키 전사법으로 입력한 소스를 XELATEX으로 처리하게 하는 폰트 매핑을 제공한다.

여러 종류의 아랍어를 지원하는 폰트를 IRMUG[15]이 제공한다. http://wiki.irmug.org/index.php/X_Series_2.

히브리어를 위한 패키지는 없다. polyglossia의 히브리어 지원이 충분해서 불필요하기 때문이다. 그러나 유니코드 히브리어 폰트가 필요하다. SBL Hebrew는 비상업적 이용이 가능한 자유폰트이며 http://www.sbl-site.org/educational/biblicalfonts.aspx에서 얻을 수 있다. Open Font 라이선스로 이용가능한 Ezra SIL 폰트도 있다. http://www.sil.org/computing/catalog/show_software.asp?id=76에서 얻을 수 있다.

script를 잘 지정해야 한다는 점을 기억하자.

```
\newfontfamily\hebrewfont[Script=Hebrew]{SBL Hebrew}
\newfontfamily\hebrewfont[Script=Hebrew]{Ezra SIL}
```

중국어, 일본어, 한국어 (CJK)

xeCJK[24] 패키지가 이 언어들에 대한 폰트 선택과 문장부호 등을 다룬다.

[13] [역주] 아랍어 알파벳의 문자 사이 연결부를 길게 늘려 식자하는 모양을 말한다.
[14] Supreme Council of Information and Communication Technology
[15] Iranian Mac User Group

2.5.2 한글과 한국어 문서

글쓴이: 역자가 한국어판을 위해 추가

현대적 텍 엔진인 XeTeX과 LuaTeX에서 한글을 식자하려면 fontspec 패키지와 한글 글리프를 가진 글꼴만 있으면 가능하다. TeX Live에 포함된 한글 트루타입 폰트 "은 글꼴"이 있으므로

```
\fontspec{UnBatang.ttf} 한글 漢字
```

와 같이 식자할 수 있다. 한자에 대해서도 마찬가지이다.

polyglossia 패키지는 korean 언어 모듈을 가지고 있어서 한글을 지원한다. 글꼴을 잘 설정해주는 것에 주의하면 된다.

```
\usepackage{polyglossia}
\setotherlanguage{korean}
\newfontfamily\hangulfont{UnBatang.ttf}
```

이제 korean 환경이나 `\textkorean` 명령을 사용할 수 있다. 원한다면 main language를 korean으로 설정하고 쓸 수도 있으며 이 경우에는 자동 생성 문자열, 행 간격, 일부 문장부호 등을 한글 문서에 알맞도록 설정해준다.

앞선 절에서 소개하는 xeCJK 패키지로 한글을 식자하는 것도 어렵지 않다. 이 때 주의할 것은 중국어나 일본어에는 띄어쓰기가 없는 데 반해 우리 글은 띄어쓰기를 한다는 점이다. 그러므로 폰트 설정 시에 이 사실을 명시해야 할 것이다.

```
\usepackage{xeCJK}
\xeCJKsetup{%
  CJKspace=true,%
  CJKecglue={}%
}
\setCJKmainfont{NanumMyeongjo}%
  [Ligatures=TeX,BoldFont={* ExtraBold},AutoFakeSlant]
```

이상의 방법들은 한국어를 주요 언어로 하는 문서보다는 보조 언어로서 한글 표기가 필요할 경우에 쓸 수 있다.

koTeX

한국어를 주요 언어로 사용하고 한글로 문서를 작성한다면 kotex 패키지군을 이용하는 것이 가장 바람직하다. 이 패키지군의 장점을 열거하면 다음과 같다.

1. TeX Live에 포함되어 있어 별도의 설치가 필요없다.
2. 다양한 텍 엔진에 대응한다. (pdfLaTeX, XeLaTeX, LuaLaTeX, plain TeX 등)
3. 자동조사(29페이지를 보라)를 비롯하여 한국어를 올바로 식자하기 위한 기능들을 갖추고 있다.
4. KTS(한국텍학회)가 공식적으로 지원하는 패키지로서 KTUG(http://www.ktug.org)을 통하여 사용상의 도움을 얻을 수 있다.

5. 단순한 문서에서 복잡한 문서나 단행본까지 kotex으로 작성된 많은 예들이 있다.
6. 확장된 기능(옛한글, CJK 언어지원, index 생성 유틸리티 등)과 클래스를 제공한다.

```
\usepackage{kotex}
```

kotex 패키지는 현재 엔진에 따라 동작이 조금씩 달라진다. 여기서는 XeLaTeX 엔진을 중심으로 기술한다. XeLaTeX이 이른바 '레거시 텍'과 가장 크게 다른 점은 시스템에 설치된 트루타입과 오픈타입 폰트를 LaTeX 문서에서 사용할 수 있다는 점이다. koTeX도 한글 폰트를 자유롭게 사용할 수 있다. XeLaTeX에서 한글 문서의 폰트 문제에 대하여 77페이지의 4.7.2절에 약간의 정보를 추가해두었다. 별다른 설정이 없으면 한글 본문글꼴로 '나눔명조'가 사용된다.

```
\usepackage[hangul]{kotex}
```

kotex에 [hangul] 옵션을 부여하면 단순히 한글을 식자하는 것을 넘어서 한글 문서에 필요한 요소들을 좀더 풍부하게 갖추어준다. 예를 들면 행 간격이 보기 좋을 정도로 늘어나고 목차의 제목이 "Contents"가 아니라 "차례"로 바뀌며 book이나 report의 \chapter가 "제 1 장"과 같은 형식으로 식자된다.

```
\documentclass{oblivoir}
```

kotex 패키지군 중에 한글 문서를 위한 클래스가 제공된다. oblivoir 클래스를 사용하면 \usepackage{kotex} 없이 바로 한글 문서를 작성할 수 있고 한글 문서에 적합한 설정을 기본적으로 갖추어준다. 이 패키지는 memoir를 바탕으로 작성되었으므로 memoir의 다양한 기능을 문서에서 사용할 수 있다.

레거시 텍의 한글 패키지

유니코드 텍 엔진이 보편화되기 전에는 한글을 표현하는 것이 쉽지 않은 일이었다. 현재 레거시 텍을 위한 한글 패키지 가운데 중요한 것은 다음 세 가지이다.

cjk-ko kotex 패키지군의 일부이지만 CJK 패키지를 활용하도록 되어 있는 간단한 패키지이다.

kotex-utf kotex 패키지군의 일부로서 이전 HLaTeX을 계승하고 있는 것이다.

CJK 중국어, 일본어, 한국어 외에도 몇 가지 아시아 언어를 더 표현할 수 있는 패키지이다.

한글을 식자하는 폰트는 cjk-ko와 kotex-utf가 nanumtype1 또는 unfonts-type1을 활용하며 CJK는 uhc 폰트를 쓴다. 이 이외에 쓸 수 있는 폰트는 거의 없다.

현재도 pdfLaTeX을 활용해야 할 경우가 적지 않다. 그럴 적에 이 패키지들을 활용할 수 있다. 만약 \usepackage{kotex} 문장이 있다면 기본적으로 kotex-utf가 실행되게 되어 있다. 레거시 텍의 kotex을 쓸 적에는 inputenc와 함께 쓰지 않도록 주의하라.

2.6 단어 사이의 공백

조판 결과 양끝맞춤이 제대로 되도록 하려고 LaTeX은 단어 사이에 가변적인 공백을 넣는다. 문장의 끝에는 텍스트의 가독성을 높이기 위해 약간 더 많은 공백을 추가한다. LaTeX은 마침표, 물음표, 느낌표가 오면 문장이 끝난 것으로 간주한다. 대문자 다음에 오는 마침표는 보통 약어를 나타내기 위한 것으로 문장의 종지로 취급하지 않는다.

이 가정에 예외가 되는 것들은 문서작성자가 그 사실을 알려주어야 한다. 스페이스 앞에 백슬래시를 두면 이 공백의 폭을 변하지 않게 하는 것이다. 틸데 문자 '~'는 폭을 고정시키면서 줄나눔도 일어나지 않도록 하는 공백을 만든다. 마침표 앞의 \@ 명령은 이 마침표가 대문자 뒤에 나온 경우라도 문장의 끝임을 표시한다.

```
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?

문장의 종지 뒤에 추가 공백이 붙지 않도록 하려면

```
\frenchspacing
```

이라고 선언한다. LaTeX으로 하여금 마침표 뒤에 추가 공백을 삽입하지 말고 일반 문자처럼 취급하라고 지시하는 것이다. 비영어권 언어에서 (문헌목록은 예외) 일반적이다. \frenchspacing을 사용한다면 \@ 명령은 불필요하다.

2.7 표제와 장절

독자의 독서를 용이하게 하기 위해 문서를 장, 절, 소절로 구분한다. LaTeX에서는 절 표제를 인자로 취하는 특별한 명령으로 이를 표현한다. 올바르게 사용하는 것은 전적으로 문서작성자의 몫이다.

다음 장절 명령은 article 클래스에서 사용할 수 있다.

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

장과 절의 번호매김을 바꾸지 않으면서 문서를 1부, 2부 등으로 구분하려면 \part 명령을 쓴다.

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

report나 book 클래스로 작업하고 있다면 추가적인 상위 장절명령이 하나 더 있다.

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

article 클래스는 chapter가 없기 때문에 article을 모아서 chapter로 묶어 책으로

만들기 쉽다. 장절 표제의 간격, 번호매김, 폰트 크기 등은 클래스의 정의에 따라 LaTeX이 자동으로 설정한다.

조금 특별한 장절명령이 두 가지 있다.

- `\part` 명령은 장 번호에 영향을 주지 않는다.
- `\appendix` 명령은 인자 없이 쓰인다. 장 번호 모양을 숫자가 아닌 문자로 바꾼다.[16]

LaTeX은 컴파일의 직전 단계에서 기억한 각 장절의 표제와 페이지 번호를 모아서 목차를 만든다.

`\tableofcontents`

이 명령이 주어진 위치에 목차를 넣는다. 새 문서라면 두 번 컴파일("LaTeX 실행")해야 올바른 목차를 생성한다. 세 번 컴파일해야 하는 경우도 있는데 추가 컴파일이 필요한지 여부를 LaTeX이 알려준다.

위에 열거한 장절 명령에는 "별표 붙은" 명령도 있다. "별표 붙은" 명령이란 명령 이름 뒤에 *를 붙여서 지시하는 것을 말한다. 이 경우에는 절 제목이 목차에 나타나지 않으며 절 번호도 붙지 않는다. 예를 들면 `\section{Help}` 대신 `\section*{Help}`로 하는 것이다.

대개 장절의 표제는 입력된 그대로 목차에 나타난다. 그런데 그 표제가 너무 길어서 목차에 넣기에는 적절치 않을 때가 있다. 목차에 넣을 (짧은) 표제를 별도로 지정하려면 실제 표제 앞에 옵션 인자로 주면 된다.

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

문서 전체의 타이틀을 만드는 명령은 다음과 같다.

`\maketitle`

타이틀을 만들기 위해 필요한 내용을 다음 명령으로 정의한다.

`\title{...}`, `\author{...}`, `\date{...}` (생략가능)

`\maketitle`을 부르기 전에 이 명령이 실행되어 있어야 한다. `\author` 명령으로 여러 사람을 열거하려 할 때는 `\and` 명령으로 각각을 구분한다.

8페이지의 그림 1.2에 위에 언급한 명령이 사용된 사례를 볼 수 있다.

위에 설명한 장절 명령과는 별도로 LaTeX 2_ε에서 `book` 클래스에서 쓰는 세 가지 명령이 추가되었다. 출판물을 구획하는 데 유용한 명령들이다. 실제 출판된 단행본에서 볼 수 있는 것처럼 장의 표제나 페이지 번호 등을 변경한다.

`\frontmatter` 이 명령은 문서 본문의 시작(`\begin{document}`) 직후에 제일 먼저 와야 한다. 페이지 번호 모양을 로마 숫자로 바꾸고 장절 표제에는 별표 붙은 명령을 준 것처럼 번호가 붙지 않는다. 그러나 목차에는 표제가 나타날 것이다.

[16] article 스타일에서는 장 번호가 아니라 절 번호가 바뀐다.

\mainmatter 이 명령은 책 본문의 첫 장 직전에 온다. 페이지 번호 모양이 아라비아 숫자로 매겨지고 페이지 번호를 1부터 새로 시작한다.

\appendix 책의 부록 부분의 시작임을 표시한다. 이 명령 이후 chapter들은 번호가 알파벳 문자로 붙는다.

\backmatter 책의 마지막 부분 바로 앞에 두는 명령이다. 이 뒤로는 참고문헌 목록, 색인 등이 온다. 표준 클래스에서 특별한 모양의 변화는 없다.

2.8 교차참조

단행본, 보고서, 논문 등에는 그림, 표, 텍스트의 특정 부분에 대하여 교차참조하는 경우가 많다. LaTeX은 교차참조를 위하여 다음과 같은 명령을 마련하고 있다.

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

여기서 *marker*는 문서작성자가 지시하는 식별자이다. `\ref` 명령을 만나면 LaTeX은 대응하는 `\label` 명령이 위치한 곳에 있는 절, 소절, 그림, 표, 정리(theorem) 들의 번호를 취하여 그 자리에 식자한다. `\pageref`은 그 `\label`이 있는 페이지의 번호를 인쇄한다.[17] 표제 텍스트와 페이지 번호로 목차를 만들 때와 마찬가지로 이 숫자(번호)들은 직전 컴파일 때에 저장해둔 것을 사용하므로 교차참조 숫자가 잘 나타나려면 두 번 이상의 컴파일이 필요하다.[18]

```
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 29.”

2.9 조사의 선택

글쓴이: 한국어판을 위하여 역자가 추가

한국어의 조사 중에는 앞말의 끝소리에 따라 형태가 교체되는 것이 있다. 한국어 문서를 작성할 때 교차참조 기능을 사용하게 되면 문서작성자가 “`\ref{<label>}`을”이라고 적는 시점에서 `\ref`가 만들어내는 숫자가 얼마일지 알 수 없다. 그러므로 그 뒤에 올 조사가 ‘을’이 될지 ‘를’이 될지를 나중에 만들어진 숫자를 보고 수정해야 한다. 이런 불편을 없애고 조사의 형태를 LaTeX이 직접 선택할 수 있게 해주는 기능이 koTeX에 있다. 만약 koTeX을 사용하고 있다면 위의 사례는 다음과 같이 백슬래시를 붙여서 입력한다.

```
\ref{autojosa}\을 보아라
```

2.9를 보아라

[17] 이 명령이 가리키는 바가 무엇인지를 명령 자체가 알고 있는 것이 아니라는 점을 기억하자. `\label`은 단지 마지막에 얻은 번호를 저장할 뿐이다.

[18] [역주] “방정식 (1)”과 같은 모양을 만드는 특별한 참조 명령 `\eqref`이 있다. 이것은 수학식의 번호를 참조하는데 amsmath 패키지에서 제공하는 것이다. 다음 장의 수학식 조판에서 사용하고 있다.

이를 "자동조사"라고 부른다. 자동조사 명령은 다음과 같다.

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

짝을 이루는 조사 명령은 어떤 것을 써도 결과가 같다.

자동조사는 교차참조의 경우뿐 아니라 숫자 방식의 인용(`\cite`)에 대해서도 믿을 만하게 동작한다.

2.10 각주

각주를 달려면 다음 명령을 쓴다.

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

그러면 현재 페이지의 하단부에 각주가 인쇄된다. 각주는 문장 전체나 그 일부에 대해 붙는[19] 것이므로 쉼표나 마침표 뒤에 붙어야 한다.[20,22]

```
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.

2.11 단어의 강조

타자기를 쓰던 시절에는 중요한 단어에 밑줄을 그어서 강조했다.

`\underline{`*text*`}`

출판물에서는 타자친 원고의 밑줄이 그어진 부분을 *italic*으로 식자하여 강조임을 표시했다. 문서작성자는 밑줄이든 이탤릭이든 신경쓸 필요가 없다. 중요한 점은 LaTeX에게 이 부분이 특히 중요하여 강조로 표현되어야 한다는 것을 알려주는 것이다.

`\emph{`*text*`}`

이 명령으로 강조 어구임을 표시할 수 있다. 실제로 인자가 어떻게 인쇄되는가는 맥락에 따라 달라진다. 다음 보기를 보라.[23]

[19] 각주를 붙인 예

[20] 각주는 문서의 본문을 읽는 독자의 주의를 다른 곳으로 돌리게 한다는 점을 기억하라. 아무튼 각주까지 모두 읽는 우리 같은 사람은 좀 별난 축에 든다. 그러니 말하고 싶은 것이 있으면 모두 본문에 포함시키는 것이 좋지 않을까?[21]

[21] 각주에 다시 각주를 붙여본 예. `\footnote`를 각주 내에 다시 쓸 수 없기 때문에 여기서는 `\footnotemark`와 `\footnotetext`를 따로 이용하였다.

[22] [역주] '미주'라 하여 페이지 바닥에 주를 달지 않고 장이나 책의 마지막에 모아두는 방법이 있다. 이를 구현하려면 endnotes 또는 enotez 패키지를 이용할 수 있다.

[23] [역주] 한글 폰트의 폰트가족에는 이탤릭체가 없다. 따라서 강조에 이탤릭을 쓰지 않는다. 종래 "우사체(右斜體)" 또는 "기울임꼴"이라는 것으로 강제로 오른쪽으로 기울어지게 변형(fake-slanted)한 글자를 이탤릭체 대용으로 쓰던 관행이 없지는 않지만 오늘날 적절한 방법으로서 추천하기는 어렵다. 한글로 된 문장에서 일부 단어를 강조하기 위해서 글자 위에 점을 찍거나 서체를 바꾸는 방법이 바람직하다.

```
\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.*

폰트의 종류나 크기를 바꾸고 싶다면 99페이지의 6.2절이 약간 도움이 될 것이다.

2.12 환경

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

이 예제에서 *environment*는 환경의 이름이다. 하나의 환경 안에 또다시 환경이 올 수도 있다. 다만 열고 닫는 순서를 잘 지켜야 한다.

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

이하 소절에서 중요한 환경을 설명하겠다.

2.12.1 리스트 문단(Itemize, Enumerate, Description)

`itemize`는 단순 리스트, `enumerate`는 번호붙은 리스트, `description`은 설명형 리스트를 위한 환경이다. 각 항목은 `\item`으로 지시해야 하며 하나 이상의 `\item`이 반드시 있어야 한다.

```
\flushleft
\begin{enumerate}
\item You can nest 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 nest 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.

2.12.2 문단의 정렬(Flushleft, Flushright, Center)

`flushleft`와 `flushright` 환경은 문단을 왼쪽 또는 오른쪽으로 몰아서 정렬해준다. `center` 환경은 텍스트를 가운데정렬한다. `\\`로 줄나눔을 강제하지 않으면 LaTeX이 자동으로 줄을 나눈다.

```
\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

2.12.3 인용문과 운문(Quote, Quotation, Verse)

quote는 인용문, 중요 구절, 예시 문장을 위해 사용할 수 있는 환경이다.

```
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 가지 더 있다. 하나는 quotation이고 다른 하나는 verse이다. quotation 환경은 인용문이 몇 단락 이상을 포함할 정도로 길 때 사용할 수 있다. 인용문의 각 단락은 들여쓰기를 한다.

verse 환경은 시와 같이 줄나눔이 중요한 운문을 조판하는 데 유용하다. 행(行)의 끝은 \\로 표시하며, 연(聯)과 연 사이에 빈 줄을 둔다.

```
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.

2.12.4 요약문(Abstract)

과학문헌은 독자에게 내용에 대한 개관을 제공하는 요약문을 앞에 두는 것이 관례이다. LaTeX에서는 `abstract` 환경을 사용하여 요약문을 만들 수 있다. 일반적으로 `abstract`는 article 클래스를 사용하는 문서에서 쓰인다.

```
\begin{abstract}
The abstract abstract.
\end{abstract}
```

Abstract

The abstract abstract.

2.12.5 그대로 보이기(Verbatim)

`\begin{verbatim}`와 `\end{verbatim}`으로 둘러싸인 부분은 마치 타자기로 친 것과 같이 줄나눔, 공백 등이 입력된 그대로 나타나며 그 안에 있는 LaTeX 명령이 전혀 실행되지 않는다.

한 문단 안에서 다음과 같이 하면 비슷한 결과를 얻을 수 있다.

`\verb+`*text*`+`

+는 시작과 끝을 나타내는 경계문자의 한 예시이다. 문자(letter), *, 공백(space)을 제외한 다른 글자를 사용할 수 있다. 이 문서의 많은 LaTeX 코딩의 예들이 이 명령을 이용해서 작성되었다.

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

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

The `\ldots` command ...

```
10 PRINT "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
```

`\verb` 명령은 별표 붙은 명령의 형태로 사용할 수 있다. 어떤 점이 다른지는 다음 예를 보라.

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

```
like␣␣␣␣this␣:-)␣
```

`verbatim` 환경과 `\verb` 명령은 다른 명령의 인자 안에서는 쓸 수 없다.

2.12.6 표(Tabular)

tabular 환경은 표를 아름답게 조판하는 데 사용한다. 표에 가로선이나 세로선을 그을 수도 있다. 컬럼의 폭은 자동으로 결정된다.

`\begin{tabular}[pos]{table spec}`

여기서 *table spec* 인자는 표의 형태를 결정하는 명령이다. `l`은 해당 열(column)의 텍스트가 왼쪽정렬 되도록 하라는 것이고 `r`은 오른쪽정렬 하라는 것이다. 가운데정렬은 `c`로 지시한다. `p{width}`는 컬럼의 내용을 *width*에 맞추어 줄나눔하고 양끝정렬하라는 것을 의미한다. `|`는 세로줄이다.

컬럼에 놓인 텍스트가 너무 길어서 페이지 폭에 넘치는 경우라도 LaTeX이 그것을 자동으로 자르지 않는다. `p{width}`를 지정하여 해당 컬럼에 오는 텍스트를 일반 문단처럼 정렬되도록 할 수 있다.

pos 인자는 인접 텍스트의 베이스라인에 대하여 표가 놓일 수직 위치를 지정한다. `t`, `b`, `c` 가운데 하나를 쓸 수 있는데 각각 top, bottom, center를 의미한다.

tabular 환경 안에서 `&`는 다음 컬럼으로 이동을 표시한다. `\\`는 줄을 나누라는 뜻이며 `\hline`은 가로선을 긋게 하는 것이다. 일부 컬럼에만 걸치는 부분 가로선을 그으려면 `\cline{`*i*`-`*j*`}`와 같이 하는데 이 때 *i*와 *j*는 부분 가로선이 그어질 범위에 해당하는 컬럼의 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.

컬럼구분자는 `@{...}` 형식으로 지시할 수도 있다. 이 명령은 컬럼 사이의 공백을 없애고 그 부분을 중괄호 사이에 온 것으로 대체한다. 이 명령의 사용례가 아래 소수점 정렬 문제를 설명하는 데서 사용되고 있다. 그리고 표의 여분 공백을 제거하기 위하여 `@{}`를 사용하는 방식으로 응용가능하다.

```
\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

아무런 패키지를 쓰지 않은 LaTeX 자체만으로는 소수점을 기준으로 컬럼의 숫자를 정렬할 방법이 없다.[24] "변칙적인" 방법을 생각해보자. 두 개의 컬럼을 사용하되 왼쪽 컬럼에는 정수부분을, 오른쪽 컬럼에는 소수부분을 넣어두는 것이다. 그런 다음 `@{.}`를 이용하여 컬럼 사이의 스페이스를 점(".")으로 대체하도록 `\begin{tabular}`의 인자를 지정하면 마치 두 컬럼이 소수점으로 정렬된 하나의 컬럼처럼 보이게 할 수 있다. 정수부분 컬럼과 소수 부분 컬럼 사이에 컬럼 분리 문자(`&`)가 있어야 한다는 점을 명심하자. 컬럼의 제목줄은 `\multicolumn` 명령을 이용하여 표시한다.

```
\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!

`tabular` 환경으로 식자되는 표는 항상 한 페이지 안에 놓인다. 표가 길어서 몇 페이지에 걸쳐 나누어져야 한다면 longtable 패키지의 같은 이름의 환경을 사용할 수 있다.

가끔 LaTeX이 그린 표가 너무 협소하게 느껴진다면 `\arraystretch`와 `\tabcolsep` 값을 조정하여 넉넉하게 만들 수 있다.

```
\begin{tabular}{|l|}
\hline
These lines\\\hline
are tight\\\hline
\end{tabular}

{\renewcommand{\arraystretch}{1.5}
\renewcommand{\tabcolsep}{0.2cm}
\begin{tabular}{|l|}
\hline
less cramped\\\hline
table layout\\\hline
\end{tabular}}
```

These lines
are tight

less cramped
table layout

[24] 'tools' 패키지묶음에 포함되어 있는 dcolumn 패키지가 이런 일을 해준다.

표의 특정 행(row)의 높이를 임의로 증가시키고 싶다면 보이지 않는 세로선[25]을 넣으면 된다. 폭이 0인 \rule을 그려서 이것을 구현해보자.

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

여기에 쓰인 pt와 ex는 TEX의 길이 단위이다. 105페이지의 표 6.5를 보라.

tabular 환경을 확장하는 많은 추가적인 명령이 booktabs 패키지에 정의되어 있다. 적절한 간격을 갖춘 전문적인 품위의 표를 상당히 쉽게 그리게 해준다.[26]

2.13 그림 포함하기

떠다니는 개체에 그림을 넣는 명령은 D. P. Carlisle이 작성한 graphicx 패키지에 의해 잘 마련되어 있다. 이 패키지는 "graphics" 패키지 묶음의 일부이다.[27] "떠다니는 개체"에 대해서는 2.14절에서 설명할 것이다.

다음 과정을 차례대로 따라해보기 바란다.

1. 그래픽 프로그램에서 그림을 EPS, PDF, PNG, JPEG 중 하나의 포맷으로 내보내기(export)한다.
2. 그림을 EPS 벡터 그래픽으로 저장하였다면 미리 PDF 포맷으로 변환해둔다. 명령행 프로그램 epstopdf을 이용하여 변환할 수 있다. 물론 PDF로 내보내기하여도 좋다. 다만 EPS로 저장한 후에 변환하는 것이 더 나을 때가 있는데 소프트웨어가 PDF로 저장할 수 있지만 그 PDF가 페이지 전체 크기로 작성되어서 문서에 들여오면 실제 그림이 아주 작아져버리는 경우가 가끔 있기 때문이다. 이럴 때는 바운딩박스를 잘 갖추고 있는 EPS로 내보낸 다음 변환하는 것이 실제 크기의 그림을 얻을 수 있는 한 방법이 된다.
3. graphicx 패키지를 전처리부에 로드하도록 지시한다.

   `\usepackage{graphicx}`

4. 문서의 본문에서 그림을 삽입할 위치에 다음 명령을 쓴다.

   `\includegraphics[`*key*=*value*, ...`]{`*file-name*`}`

   선택 인자는 *key*와 이에 연결된 *value*의 리스트를 쉼표로 구분하여 받아들인다. *key*에 width, height, rotation이 올 수 있다. 이 값들은 삽입할 그림의 너비, 높이, 회전을 나타낸다. 표 2.2에 중요한 key를 요약해놓았다.

그림 2.2의 예시 코드를 보면 이해가 쉬울 것이다. 이 코드는 test.png라는 이름의

[25] 전문 조판에서 이것을 strut라고 한다.

[26] [역주] booktabs 패키지는 세로선이 없고 상하 가로선에 다른 두께를 주는 표를 그리는 데 유용하다. 이런

표 2.2: graphicx 패키지의 key

`width`	그림의 너비를 주어지는 값으로 맞춘다.
`height`	그림의 높이를 주어지는 값으로 맞춘다.
`angle`	그림을 반시계방향으로 주어진 각도만큼 회전시킨다.
`scale`	그림을 비례적으로 늘리거나 줄인다.

```
\includegraphics[angle=90,width=.5\textwidth]{test.png}
```

그림 2.2: `test.png`를 문서에 삽입하는 예시 코드

그림을 포함하는 것이다. 그림을 **먼저** 90도 회전시키고 **그 다음에** 표준 문단 폭의 0.5배를 최종적인 너비로 설정하여 그림을 축소(확대)한다. 가로세로비는 1.0이다. 왜냐하면 높이(height) 값이 별도로 주어지지 않았기 때문이다. width와 height 인자의 값으로는 절대 길이를 지정할 수도 있다. 길이에 대한 더 자세한 사항은 105페이지의 표 6.5를 보라. 그림 삽입에 관하여 더 알고 싶으면 [9]를 꼭 읽어보자.

2.14 떠다니는 개체

오늘날 대부분의 출판물에는 많은 그림과 표가 포함되어 있다. 이러한 요소들은 특별한 취급이 필요한데 그 이유는 이들을 잘라서 다른 페이지에 배치할 수 없기 때문이다. 한 가지 방법은 그림이나 표가 현재 페이지의 남은 공간에 다 들어가지 않을 때마다 새로운 페이지를 시작하는 것인데 이렇게 하면 페이지에 빈 공간이 생기게 되고 결과적으로 보기 좋지 않다.

이 문제에 대한 해결책은 현재 페이지 남은 부분에 배치할 수 없는 그림이나 표를 "떠다니게" 하고 페이지의 남은 부분에 텍스트가 연속되어 흐르게 하는 것이다. LaTeX은 이러한 떠다니는 개체를 위한 두 가지 환경을 마련하고 있다. 하나는 표를 위한 것이고 다른 하나는 그림을 위한 것이다. 이 두 가지 환경을 잘 활용하려면 LaTeX이 떠다니는 개체를 내부적으로 어떻게 다루는가를 대강 이해하는 것이 중요하다. 그렇지 못하면 떠다니는 개체는 그저 골칫거리가 되고 말아서 LaTeX은 원하는 위치에 그림이 나오지 않는다고 불평하게 하는 원인이 된다.

먼저 떠다니는 개체를 위한 LaTeX 명령을 살펴보자.

`figure`나 `table` 환경으로 둘러싸인 부분은 떠다니는 개체로 간주된다.

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

두 환경에 선택 인자를 줄 수 있다. 이것을 **위치지정자**(*place specifier*)라고 한다. 이 인자들은 개체를 이동시켜 놓을 위치를 LaTeX에게 알려주는 역할을 한다. **위치지정자**는 **허용위치**

유형의 표가 한글 문헌에서는 많지 않은 편이다.

[27] CTAN://pkg/graphics

를 나타내는 문자열로 이루어진다. 표 2.3을 보라.[28]

예를 들면 다음 한 줄로 table이 시작된다.

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

위치지정자 `!hbp`는 이 표가 바로 이곳(`h`)이나 페이지의 바닥(`b`) 아니면 별도 페이지(`p`)에 위치할 수 있도록 하면서 이 위치잡기가 그다지 만족스럽지 못한 경우라도(`!`) 이행하라는 의미를 가지고 있다. 위치지정자가 주어지지 않았을 때 표준 클래스에서는 `[tbp]`를 쓰도록 정해져 있다.

LaTeX은 떠다니는 개체를 만날 때마다 문서작성자가 지시한 위치지정자에 맞추어서 위치를 정한다. 한 개체가 현재 페이지에서 자리를 잡지 못하면 그것을 *figures*나 *tables* 대기열에 넣어두고 처리를 미룬다.[29] LaTeX은 대기열에 있는 개체들로 특별한 '플로트가 놓이는' 페이지를 만들기에 충분한지를 검사한다. 이것이 만약 불가능하면 대기열의 첫 번째 개체가 지금 막 그 위치에서 불린 것처럼 취급한다. 즉 LaTeX은 그 개체에 주어져 있는 위치지정자의 요구에 따라 다시 위치를 결정하려 한다. (단 'h' 지정자는 더이상 무의미하다.) 텍스트에서 떠다니는 개체가 새로이 발생할 때마다 각자 대기열 속에 들어간다. LaTeX은 개체의 유형별로 출현 순서를 엄격히 유지한다. 그러다 보면 문서의 끝까지 더이상 그림의 자리를 잡지 못하고 미루어지는 경우가 생기는 것이다. 그러므로,

> 만약 LaTeX이 그림이나 표를 원하는 곳에 출력해주지 않는다면 두 플로트 대기열 가운데 하나에서 어떤 개체가 처리되지 않은 채로 정체되어 있기 때문이다.

LaTeX에게 위치지정자를 하나만 전달하는 것도 가능하지만 이렇게 하면 문제가 생긴다. 만약 플로트가 그 지정된 위치에 놓이는 데 실패하면 바로 정체되어 그 이후에 오는 플로트들이 처리되지 못하게 만든다. 특히, [h] 옵션 하나만 지정되어 있다면 아주 곤란하기 때문에 요즘 LaTeX은 문서작성자가 [h]라고만 하더라도 자동으로 [ht]로 바꾸어서 처리한다.

표 2.3: 떠다니는 개체의 허용 위치

지정자	개체가 놓이도록 허용되는 위치 ...
`h`	*here*. 텍스트상의 현재 위치. 크지 않은 떠다니는 개체에 유용하다.
`t`	*top*. 페이지의 상단
`b`	*bottom*. 페이지의 하단
`p`	*page*. 그림과 표만으로 이루어지는 별도 페이지
`!`	내부 파라미터 값에 따르면 현재의 개체를 놓을 수 없을 때 내부 파라미터[a]를 무시하도록 함

[a]예를 들면 한 페이지에 허용되는 떠다니는 개체의 수 상한값 등

[28][역주] `table`과 `tabular`를 우리말로 둘 다 '표'라고 하다 보니 생기는 오해가 있다. `tabular`는 행과 열에 맞추어서 요소를 배열하고 필요하다면 가로선과 세로선을 그은 식자 형태를 가리키는 것인 반면, `table`은 그렇게 만들어진 `tabular`를 둘러싸서 떠다니게 만드는 환경을 가리킨다. 실제로 `table` 환경 안에 오는 내용물은 반드시 표가 아니라도 상관없고 심지어 그림이 와도 된다. 그러나 `\caption`을 붙이면 명칭이 "표 1:" 또는 "Table 1:"과 같이 나타난다. `figure`에 대해서도 같다.

[29]FIFO, 즉 "먼저 들어온 것이 먼저 나가는" 큐(queue)이다.

좀 어려운 부분을 설명하였고, table과 figure 환경에 대해 언급할 것이 몇 가지 더 있다.

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

이 명령은 떠다니는 개체에 캡션을 달아준다. "Figure" 또는 "Table"이라는 문자열[30]이 붙은 그림번호 또는 표번호가 LaTeX에 의해 부여된다.

다음 두 명령은 그림 목차와 표 목차를 생성한다.

`\listoffigures \listoftables`

이 명령은 \tableofcontents 명령과 같은 방식으로 동작하여 각각 그림과 표의 목차를 인쇄한다. 이 목차에는 캡션이 식자되는데 만약 캡션이 길어서 문제가 되면 목차에 나타날 짧은 캡션을 지정해두어야 한다. \caption 명령의 대괄호 선택 인자에 짧은 캡션을 적어두는 방식으로 하면 된다.

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

\label과 \ref는 문서 안에서 특정 플로트에 대하여 참조할 수 있게 해준다. \label 명령은 반드시 \caption 명령 **이후에** 써야 한다는 것을 기억하라. 왜냐하면 참조할 숫자(번호)가 캡션이 생성될 때 만들어지기 때문이다.

다음 보기는 사각형을 그려서 문서에 삽입하고 있다. 문서에 그림 넣을 공간을 만든 다음에 최종적으로 출력하여 그림을 풀로 붙여넣을 생각이라면 이 코드를 이용할 수 있다.

```
Figure~\ref{white} is an example of Pop-Art.
\begin{figure}[!hbtp]
\includegraphics[angle=90,width=\textwidth]{white-box.pdf}
\caption{White Box by Peter Markus Paulian.\label{white}}
\end{figure}
```

이 예에서 LaTeX은 바로 이곳 (h)에 그림을 놓으려고 **할 수 있는 한** (!) 시도해 본다.[31] 만약 이것이 불가능하면 그림을 페이지의 **바닥** (b)에 놓으려 한다. 현재 페이지에서 그것이 불가능하면 (표 대기열에 혹시 표가 있다면 그것과 함께) 이 그림을 놓을 별도의 플로트 페이지를 만들 수 있을지를 결정한다. 개체들이 페이지를 채울 수 있을 정도가 되지 않는다면 LaTeX은 새 페이지를 시작하여 이 위치에서 그림이 주어진 것처럼 한 번 더 처리 과정을 시도한다.

어떤 경우에는 다음 명령을 써야할 때가 있다.

`\clearpage 또는 \cleardoublepage`

LaTeX에게 즉시 대기열에 남아 있는 모든 떠다니는 개체를 식자하여 대기열을 비우고 새로운 페이지를 열라는 것이다. \cleardoublepage는 새로 열리는 페이지가 홀수페이지 (오른쪽 면)가 되도록 필요하다면 짝수면을 추가하여 새로운 페이지로 이동하라는 의미이다.[32]

[30] [역주] 한국어 텍에서는 각각 "그림"과 "표"로 식자된다.
[31] 대기열에 다른 그림이 없다고 가정.
[32] [역주] \newpage는 떠다니는 개체를 즉시 청산(flush)하도록 요구하지 않는다.

제 3 장

수학식 조판

이제 준비가 됐다! 이 장에서는 TEX의 가장 중요한 장점에 해당하는 수학식 조판을 공략해볼 것이다. 그렇지만 여기서 다루는 것은 수박겉핥기에 불과하다. 여기 설명하는 내용이 대부분 충분하다고 보지만 혹시 원하는 수학식 조판의 해법을 발견하지 못했다고 해서 실망할 필요는 없다. $\mathcal{AMS}$-LATEX에서 그 문제에 대한 해결책을 제공하고 있을 가능성이 높다.

3.1 $\mathcal{AMS}$-LATEX

(고급) 수학식을 조판하려면 $\mathcal{AMS}$-LATEX을 이용해야 한다. $\mathcal{AMS}$-LATEX은 수학식 조판을 위한 클래스와 패키지의 모음이다. 그 가운데서도 amsmath 패키지를 주로 검토할 것이다. $\mathcal{AMS}$-LATEX은 미국수학회가 만든 것으로 수학식 조판에 널리 쓰이고 있다. LATEX 자체가 어느 정도 수학식 조판 기능과 환경을 제공하고 있기는 하지만 한계가 있고(달리 말하자면 $\mathcal{AMS}$-LATEX에는 한계가 없다는 말이다) 일관성이 결여된 경우도 있다.

$\mathcal{AMS}$-LATEX은 필수 배포판의 일부이며 모든 LATEX 배포판에 포함되어 있다.[1] 이 장에서 논의하는 사항은 전처리부에 amsmath가 당연히 `\usepackage{amsmath}`와 같이 로드되어 있다는 것을 전제로 한다.

3.2 수식 기초

수식은 문단 가운데 문장의 일부로 식자되거나(*text style*) 별도의 문단으로 구분되어 식자되거나(*display style*) 한다. 앞의 것을 행중수식(in-line), 뒤의 것을 별행수식(display)이라 한다. 행중수식은 $와 $ 사이에 입력한다.

[1] 자신의 시스템에 이게 설치되어 있지 않다면 CTAN://pkg/amslatex으로 가보라.

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

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

```
\TeX{} is pronounced as
$\tau\epsilon\chi$\\[5pt]
100~m$^{3}$ of water\\[5pt]
This comes from my $\heartsuit$
```

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

100 m^3 of water

This comes from my ♡

좀 큰 수식을 문단과 구분하여 별행으로 식자하고자 한다면 텍스트 문단을 나누어서 입력할 것이 아니라 별행수식으로 하는 것이 좋다. 한 줄 수식을 별행으로 식자하려면 \begin{equation}과 \end{equation}으로 둘러싼다.[2] \label을 붙여서 수식의 번호를 표시하고 다른 곳에서 이 수식번호를 \eqref 명령으로 참조할 수 있다. 수식 이름을 별도로 특정하려 한다면 \tag 명령을 쓴다.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach
 \begin{equation}
   a^2 + b^2 = c^2
 \end{equation}
Einstein says
 \begin{equation}
   E = mc^2 \label{clever}
 \end{equation}
He didn't say
 \begin{equation}
  1 + 1 = 3 \tag{dumb}
 \end{equation}
This is a reference to
\eqref{clever}.
```

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

$$a^2 + b^2 = c^2 \tag{3.1}$$

Einstein says

$$E = mc^2 \tag{3.2}$$

He didn't say

$$1 + 1 = 3 \tag{dumb}$$

This is a reference to (3.2).

수식 번호가 붙지 않게 하려면 별표붙은 equation* 환경을 쓰거나 훨씬 간단한 \[, \]로 둘러싸거나 한다.[3]

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach
 \begin{equation*}
   a^2 + b^2 = c^2
 \end{equation*}
or you can type less for the
same effect:
 \[ a^2 + b^2 = c^2 \]
```

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

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

or you can type less for the same effect:

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

\[가 짧고 좋기는 해도 equation과 equation*처럼 간단히 번호를 붙이거나 붙이지 않거나 하는 것은 할 수 없다.

[2] 이 환경은 amsmath 명령이므로 어떤 이유에선가 이 패키지를 로드하지 못했다면 LATEX의 별행수식 환경인 displaymath를 쓸 수 있다.

[3] 이것 역시 amsmath 방식이다. 표준 LATEX에는 별표붙은 equation 환경이 없다.

행중수식의 text style과 별행수식의 display style의 조판 스타일 차이가 무엇인지 확인하자.

```
This is text style:
$\lim_{n \to \infty}
 \sum_{k=1}^n \frac{1}{k^2}
 = \frac{\pi^2}{6}$.
And this is display style:
 \begin{equation}
  \lim_{n \to \infty}
  \sum_{k=1}^n \frac{1}{k^2}
  = \frac{\pi^2}{6}
 \end{equation}
```

This is text style: $\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$. And this is display style:

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6} \tag{3.3}$$

텍스트 스타일에서 위나 아래로 튀어나오는 수식 부분을 횡으로 늘어놓거나 첨자를 \smash하거나 큰 수식기호를 조금 작은 것으로 사용하여 수식 표현의 높이를 행 높이에 맞춤으로써 행 사이의 간격을 가지런하게 유지하려 한다.[4]

```
A $d_{e_{e_p}}$ mathematical
expression  followed by a
$h^{i^{g^h}}$ expression. As
opposed to a smashed
\smash{$d_{e_{e_p}}$} expression
followed by a
\smash{$h^{i^{g^h}}$} expression.
```

A $d_{e_{e_p}}$ mathematical expression followed by a $h^{i^{g^h}}$ expression. As opposed to a smashed $d_{e_{e_p}}$ expression followed by a $h^{i^{g^h}}$ expression.

3.2.1 수식 모드

수식 모드는 텍스트 모드와 차이가 있다. 예를 들면 **수식 모드**에서는

1. 빈 칸(스페이스)이나 줄바꿈을 입력해도 아무런 의미가 없다. 수식에서 간격은 수학식 표현의 논리상 자동적으로 설정되는 것이 아니면 \,, \quad, \qquad 같은 특별한 간격 명령으로 지정해야 한다.
2. 빈 줄은 허용하지 않는다. 모든 수학식은 수식 하나가 한 단락이다.
3. 입력한 글자는 변수 이름으로 간주되어 변수로서 조판될 것이다. 수식 안에 일반 텍스트를 (곧게 선 폰트로 띄어쓰기를 하여서) 식자하려면 \text{...} 명령을 사용하여 입력하여야 한다. (99페이지의 6.2절도 참고하라.)

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

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

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

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

[4] [역주] LaTeX 처음사용자가 부딪치는 문제 가운데 하나로 텍스트 스타일로 문단 중에 놓인 수식이 "어색해보인다"는 것이 있다. 수학교과서와 같이 조판상의 규칙을 지키는 것보다 수식 자체를 명료하게 전달하는 것이 더 중요한 책이 눈에 익어 있기 때문에 느끼는 것일 수 있다. 원한다면 행중에서도 displaystyle의 수식을 쓰도록 할 수 있으나 이 선택은 전적으로 북 디자이너의 전문적 식견에 맡겨야 하는 일일 것이다.

수학자들은 기호 사용에 대해 매우 까다롭다. 위의 예에서 $\mathbf{R}$이라고 한 것을 '블랙보드 볼드'체로 쓰는 것을 좋아할 수도 있는데 이 기호는 amssymb 패키지의 `\mathbb` 명령으로 식자할 수 있다. 그러면 위의 예제는 다음과 같이 된다.

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

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

더 많은 수학 글꼴에 대해서 63페이지의 표 3.14를 참고하라.

3.3 수식 구성하기

이 절에서 수식 조판 명령 중 가장 요긴한 것을 서술한다. 여기서 언급하는 명령은 대부분 amsmath를 필요로 하지 않는다. (만약 필요하다면 명시적으로 밝히겠다.) 그렇지만 로드해두는 것이 좋다.

그리스 문자의 소문자는 `\alpha`, `\beta`, `\gamma`, ...와 같이, 대문자는 `\Gamma`, `\Delta`, ...등과 같이 입력한다.[5] 그리스 알파벳 목록은 60페이지의 표 3.2를 보라.

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

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

지수, 위첨자, 아래첨자는 ^와 _ 부호로써 표시한다. 수식 명령은 대부분 직후 문자에만 효력을 가진다. 그러므로 어떤 명령이 여러 문자에 영향을 주도록 하려면 중괄호 `{...}`를 이용하여 그룹으로 묶어주어야 한다.

```
$p^3_{ij} \qquad
 m_\text{Knuth}\qquad
\sum_{k=1}^3 k \\[5pt]
 a^x+y \neq a^{x+y}\qquad
 e^{x^2} \neq {e^x}^2$
```

$$p_{ij}^3 \qquad m_{\text{Knuth}} \qquad \sum_{k=1}^3 k$$
$$a^x + y \neq a^{x+y} \qquad e^{x^2} \neq e^{x2}$$

예제에 나오는 $\neq$와 같은 이항관계연산부호 명령을 61페이지 표 3.3에서 $\subseteq$, $\perp$ 등 여러 가지 볼 수 있다.

제곱근은 `\sqrt`로 입력한다. n제곱근을 표시하려면 `\sqrt[`n`]`으로 쓴다. 제곱근 부호의 크기는 자동으로 결정된다. 근호의 앞머리만 필요하다면 `\surd` 명령이 있다.
예제에 나오는 $\Leftrightarrow$와 같은 화살표를 62페이지의 표 3.6에서 $\hookrightarrow$, $\rightleftharpoons$ 등 여러 종류 볼 수 있다.

```
$\sqrt{x} \Leftrightarrow x^{1/2}
 \quad \sqrt[3]{2}
 \quad \sqrt{x^{2} + \sqrt{y}}
 \quad \surd[x^2 + y^2]$
```

$$\sqrt{x} \Leftrightarrow x^{1/2} \quad \sqrt[3]{2} \quad \sqrt{x^2 + \sqrt{y}} \quad \surd[x^2 + y^2]$$

곱셈을 나타내는 점은 보통 표시하지 않는 편이 식을 한눈에 알아보기 쉬울 때가 있다. 가운뎃 **점** 하나를 찍으려면 `\cdot`을 쓴다. `\cdots`는 가운데 위치에 점을 세 개 찍고 `\ldots`는 베이스라인에 찍는다. 그밖에 `\vdots`는 세로로, `\ddots`는 대각선 방향으로 점 세 개를 놓는다. 3.6절에 더 많은 예제가 있다.

[5] 대문자 알파, 대문자 베타 등에 해당하는 명령은 따로 없다. 이 글자들은 로마자 A, B와 같은 모양이어서 별도의 명령을 만들지 않았다.

```
$\Psi = v_1 \cdot v_2
 \cdot \ldots \qquad
 n! = 1 \cdot 2
 \cdots (n-1) \cdot n$
```

$$\Psi = v_1 \cdot v_2 \cdot \ldots \qquad n! = 1 \cdot 2 \cdots (n-1) \cdot n$$

\overline과 \underline은 식의 위나 아래에 **가로선**을 긋는다.

```
$0.\overline{3} =
 \underline{\underline{1/3}}$
```

$$0.\overline{3} = \underline{\underline{1/3}}$$

\overbrace와 \underbrace는 식의 위나 아래에 길게 **누운 괄호**를 그린다.

```
$\underbrace{\overbrace{a+b+c}^6
 \cdot \overbrace{d+e+f}^7}
 _\text{meaning of life} = 42$
```

$$\underbrace{\overbrace{a+b+c}^{6} \cdot \overbrace{d+e+f}^{7}}_{\text{meaning of life}} = 42$$

변수에 **작은 화살표**나 **틸데**와 같은 수학 액센트 부호를 붙이려면 60페이지의 표 3.1이 도움이 될 것이다. 여러 글자에 걸치는 넓은 hat이나 틸데는 \widetilde, \widehat으로 만들 수 있다. 다음 보기에서 \hat과 \widehat의 차이점 그리고 아래첨자가 있는 변수에서 \bar의 위치에 주의하라. '프라임'을 표시하려면 아포스트로피 '를 붙인다.

```
$f(x) = x^2 \qquad f'(x)
 = 2x \qquad f''(x) = 2\\[5pt]
 \hat{XY} \quad \widehat{XY}
 \quad \bar{x_0} \quad \bar{x}_0$
```

$$f(x) = x^2 \qquad f'(x) = 2x \qquad f''(x) = 2$$
$$\hat{XY} \quad \widehat{XY} \quad \bar{x_0} \quad \bar{x}_0$$

벡터를 표현하기 위해 변수 위에 작은 화살표 기호를 붙이는 경우가 있다. \vec 명령을 쓴다. \overrightarrow와 \overleftarrow 명령은 A에서 B로 가는 벡터를 표시할 때 사용할 수 있다.

```
$\vec{a} \qquad
 \vec{AB} \qquad
 \overrightarrow{AB}$
```

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

이탤릭으로 식자하는 변수 이름과 달리 함수 이름은 바로 선(upright) 글자체로 식자한다. LaTeX은 대부분의 함수 이름을 식자할 수 있도록 다음과 같은 명령을 제공한다.

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

```
\begin{equation*}
  \lim_{x \rightarrow 0}
  \frac{\sin x}{x}=1
\end{equation*}
```

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

위의 목록에 나오지 않는 함수 이름을 쓰려면 \DeclareMathOperator 명령을 정의할 수 있다. 별표 붙은 명령으로 limit형 함수[6] 이름을 정의한다. 이 명령은 전처리부에만 둘 수 있기 때문에 다음 예에서는 주석문으로 처리했는데 이 두 줄이 전처리부에 들어 있어야 한다.

```
%\DeclareMathOperator{\argh}{argh}
%\DeclareMathOperator*{\nut}{Nut}
\begin{equation*}
  3\argh = 2\nut_{x=1}
\end{equation*}
```

$$3\operatorname{argh} = 2\operatorname*{Nut}_{x=1}$$

잉여연산(모듈로)에 두 가지 명령이 있다. \bmod는 "$a \bmod b$"처럼 쓰는 이항연산자이고 \pmod는 "$x \equiv a \pmod{b}$"와 같이 표현할 때에 쓰는 것이다.

```
$a\bmod b \\
 x\equiv a \pmod{b}$
```

$$a \bmod b$$
$$x \equiv a \pmod{b}$$

분수는 \frac{...}{...} 명령으로 표시한다. 행중수식에서 분수는 행 높이에 맞추어 크기가 줄어든다. 항상 디스플레이 스타일에서도 이 모양이 유지되게 하려면 \tfrac을 쓴다. 반대로 텍스트 스타일에서도 디스플레이 스타일로 식자되게 하려면 \dfrac 명령을 사용한다.[7] 1/2처럼 빗금으로 분수를 표시할 수도 있는데 소량의 분할분을 나타내는 작은 분수라면 이것이 나을 때도 있다.

```
In display style:
\begin{equation*}
  3/8 \qquad \frac{3}{8}
  \qquad \tfrac{3}{8}
\end{equation*}
```

In display style:

$$3/8 \qquad \frac{3}{8} \qquad \tfrac{3}{8}$$

```
In text style:
$1\frac{1}{2}$~hours \qquad
$1\dfrac{1}{2}$~hours
```

In text style: $1\frac{1}{2}$ hours $1\dfrac{1}{2}$ hours

편미분을 나타내기 위한 \partial 명령이 있다.

```
\begin{equation*}
  \sqrt{\frac{x^2}{k+1}}\qquad
  x^\frac{2}{k+1}\qquad
  \frac{\partial^2f}
  {\partial x^2}
\end{equation*}
```

$$\sqrt{\frac{x^2}{k+1}} \qquad x^{\frac{2}{k+1}} \qquad \frac{\partial^2 f}{\partial x^2}$$

이항 계수(조합) 그리고 그와 유사한 구조를 식자하려면 amsmath의 \binom 명령을 사용한다.

[6] [역주] $\lim_{x\to 0}$과 같이 디스플레이 스타일에서 첨자가 아래 위로 붙는 함수

[7] [역주] 앞서 "행중수식"과 "별행수식"에 대하여 언급하였다. 행중수식을 조판하는 스타일을 "텍스트 스타일(text style)"이라 하고 별행수식을 조판하는 스타일을 "디스플레이 스타일(display style)"이라고 한다. 이 용어는 수식모드에 적용되는 것으로 "텍스트 스타일"이라고 한 것은 "수식의 텍스트 스타일"이라는 의미이다.

```
Pascal's rule is
\begin{equation*}
 \binom{n}{k} =\binom{n-1}{k}
 + \binom{n-1}{k-1}
\end{equation*}
```

Pascal's rule is

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

이항 관계연산자는 서로 겹쳐 표현해야 할 때가 있다. `\stackrel{#1}{#2}`는 `#2`를 정상적인 위치에 놓고 그 위에 위첨자 크기로 `#1`을 얹은 기호를 만들어준다.

```
\begin{equation*}
 f_n(x) \stackrel{*}{\approx} 1
\end{equation*}
```

$$f_n(x) \stackrel{*}{\approx} 1$$

적분기호는 `\int` 명령으로 만든다. **합기호**는 `\sum`, **곱기호**는 `\prod`이다. 아래 위의 리미트는 첨자의 경우와 마찬가지로 `^`와 `_`로 지시한다.

```
\begin{equation*}
\sum_{i=1}^n \qquad
\int_0^{\frac{\pi}{2}} \qquad
\prod_\epsilon
\end{equation*}
```

$$\sum_{i=1}^{n} \qquad \int_0^{\frac{\pi}{2}} \qquad \prod_{\epsilon}$$

복잡한 표현식에서 인덱스의 위치를 잘 위치짓기 위해 amsmath 패키지의 `\substacks` 명령을 쓸 수 있다.

```
\begin{equation*}
\sum^n_{\substack{0<i<n \\
        j\subseteq i}}
   P(i,j) = Q(i,j)
\end{equation*}
```

$$\sum_{\substack{0<i<n \\ j\subseteq i}}^{n} P(i,j) = Q(i,j)$$

괄호와 그밖에 온갖 유형의 **여닫는 부호**(delimiters) (예: $[\ \langle\ \|\ \updownarrow$)를 LaTeX이 제공한다. 둥근 괄호와 각진 괄호는 키보드의 해당 키로 입력할 수 있다. 활 괄호는 `\{`로 입력한다. 그밖의 여닫는 부호는 특정 명령으로 (예: `\updownarrow`) 입력한다. 다음 보기는 `{...}` 입력과 `\{...\}`의 차이를 보여준다.

```
\begin{equation*}
{a,b,c} \neq \{a,b,c\}
\end{equation*}
```

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

여닫는 부호의 여는 부호 앞에 `\left`를 붙이고 닫는 부호에 `\right`를 붙이면 LaTeX은 크기를 자동으로 조절한다. `\left`를 썼으면 반드시 그에 대응하는 `\right`가 있어야 한다는 사실에 유의하라. 여는 부호만 두고 닫는 부호를 쓰지 않을 적에라도 꼭 "`\right.`"로 짝을 맞추어주어야 한다. 닫는 부호만 있다면 `\left.`을 먼저 적고 `\right`와 닫는 부호를 입력한다.

```
\begin{equation*}
1 + \left(\frac{1}{1-x^{2}}
    \right)^3 \qquad
\left. \ddagger \frac{~}{~}\right)
\end{equation*}
```

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

여닫는 부호의 크기를 자동으로 조절하도록 하지 않고 직접 크기를 지정하는 것이 필요할 때가 있다. 이럴 때는 \big, \Big, \bigg, \Bigg 명령을 여닫는 부호의 앞에 붙이면 된다.

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

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

사용가능한 여닫는 부호의 목록을 62페이지의 표 3.8에서 볼 수 있다.

3.4 긴 수식: multline

어떤 식이 너무 길어서 행을 나누어 표시해야 할 때가 있다. 그러나 행을 나눈 수식은 그렇지 않은 것에 비해 읽고 이해하기 어렵다. 가독성을 향상하기 위해 행을 나눌 때 적용하는 규칙이 있다.

1. 일반적으로 수식은 등호나 연산자 부호 **앞에서** 행을 나눈다.
2. 다른 부호보다 등호 앞에서 나누는 것이 좋다.
3. 곱셈 부호보다는 덧셈이나 뺄셈 부호 앞에서 행을 나누는 것이 좋다.
4. 이 이외의 곳에서는 가능하면 행을 나누지 않는다.

multline 환경을 이용하는 것이 가장 쉬운 방법이다.[8]

```
\begin{multline}
  a + b + c + d + e + f
  + g + h + i
  \\
  = j + k + l + m + n
\end{multline}
```

$$\begin{multline} a + b + c + d + e + f + g + h + i \\ = j + k + l + m + n \end{multline} \tag{3.4}$$

equation 환경과 비교해보면 수식을 여러 줄로 나누어서 표현할 수 있다는 것이다. \\를 행이 나누어질 위치에 적어넣는다. equation에 별표 붙인 equation*이 있듯이 multline* 환경이 있어서 수식 번호를 붙이지 않게 한다.

IEEEeqnarray 환경(3.5절 참조)이 더 나은 결과를 보여줄 때가 있다. 다음과 같은 경우를 생각해보자.

```
\begin{equation}
  a = b + c + d + e + f
  + g + h + i + j
  + k + l + m + n + o + p
  \label{eq:equation_too_long}
\end{equation}
```

$$a = b+c+d+e+f+g+h+i+j+k+l+m+n+o+p \tag{3.5}$$

[8] multline 환경은 amsmath에 정의된 것이다.

이 식의 우변은 너무 길고 한 줄에 다 들어가지 않는다. multline 환경으로는 다음과 같은 결과를 얻는다.

```
\begin{multline}
  a = b + c + d + e + f
  + g + h + i + j \\
  + k + l + m + n + o + p
\end{multline}
```

$$a = b + c + d + e + f + g + h + i + j + k + l + m + n + o + p \quad (3.6)$$

이것이 식 (3.5)보다 낫다. 그러나 k 앞의 덧셈 기호가 두드러지게 되어 그보다 훨씬 큰 등호의 중요성이 사라져버렸다. IEEEeqnarray 환경이 더 나은 결과를 제공한다. 이 문제에 대한 자세한 토론은 이어지는 3.5절에서 하기로 한다.

3.5 여러 줄 수식

일련의 등식이 이어질 때 한 줄에 딱 들어맞지 않는 상황을 자주 접한다. 이 절에서는 여러 개의 식을 읽기 쉽고 아름답게 수직으로 정렬하는 방법에 대해서 알아보겠다.

방법을 알아보기 전에 먼저 일반적 방법의 큰 결점을 보여주는 좋지 않은 예들을 살펴보려 한다.

3.5.1 전통적 방법의 문제점

여러 줄 수식을 align 환경으로 다음과 같이 사용할 수 있다.[9]

```
\begin{align}
  a &= b + c \\
  &= d + e
\end{align}
```

$$a = b + c \quad (3.7)$$
$$= d + e \quad (3.8)$$

그런데 한 줄이 유달리 길다면 좋은 결과를 얻기 힘들다.

```
\begin{align}
  a &= b + c \\
  &= d + e + f + g + h + i
  + j + k + l \nonumber \\
  &+ m + n + o \\
  &= p + q + r + s
\end{align}
```

$$a = b + c \quad (3.9)$$
$$= d + e + f + g + h + i + j + k + l$$
$$+ m + n + o \quad (3.10)$$
$$= p + q + r + s \quad (3.11)$$

여기서 $+ m$은 등호 아래가 아니라 d 밑에 와야 맞다. TEX 전문가(TEXpert)라면 \mathrel{} \negmedspace {} 명령을 +m+n+o 앞에 두면 필요한 공간을 확보할 수 있지 않느냐고 할지 모르겠지만 대부분의 사용자들은 그렇게까지 상상력이 미치지 않는다. 더 간단한 해결방법이 있다면 좋을 것이다.

eqnarray 환경은 어떨지 시도해본다.

[9] align 환경은 여러 수식군을 서로 나란히 놓는 데도 사용할 수 있다. IEEEeqnarray 환경에서 {rCl+rCl} 인자를 주어서 이것을 멋지게 처리할 수 있다.

```
\begin{eqnarray}
  a & = & b + c \\
  & = & d + e + f + g + h + i
  + j + k + l \nonumber \\
  && +\: m + n + o \\
  & = & p + q + r + s
\end{eqnarray}
```

$$
\begin{aligned}
a &= b+c && (3.12)\\
&= d+e+f+g+h+i+j+k+l \\
&\quad +m+n+o && (3.13)\\
&= p+q+r+s && (3.14)
\end{aligned}
$$

앞서 말한 문제는 좀 나아졌지만 여전히 문제가 있다. 등호 좌우의 공백이 너무 크다. `multline`과 `equation` 환경에서는 그렇지 않았던 문제이다.

```
\begin{eqnarray}
  a & = & a = a
\end{eqnarray}
```

$$
\begin{aligned}
a &= a = a && (3.15)
\end{aligned}
$$

…… 그리고 이 방법은 이따금 여유 공간이 없을 때 수식 번호가 겹쳐나타나기도 한다.

```
\begin{eqnarray}
  a & = & b + c
  \\
  & = & d + e + f + g + h^2
  + i^2 + j + k
  \label{eq:faultyeqnarray}
\end{eqnarray}
```

$$
\begin{aligned}
a &= b+c && (3.16)\\
&= d+e+f+g+h^2+i^2+j+k && (3.17)
\end{aligned}
$$

좌변이 너무 길 때 `\lefteqn` 명령을 쓸 수 있다.

```
\begin{eqnarray}
  \lefteqn{a + b + c + d
    + e + f + g + h}\nonumber\\
  & = & i + j + k + l + m
  \\
  & = & n + o + p + q + r + s
\end{eqnarray}
```

$$
\begin{aligned}
&a+b+c+d+e+f+g+h \\
&\quad = i+j+k+l+m && (3.18)\\
&\quad = n+o+p+q+r+s && (3.19)
\end{aligned}
$$

그런데 우변이 너무 짧으면 문제가 생겨서 수식의 가운데 정렬이 실패한다.

```
\begin{eqnarray}
  \lefteqn{a + b + c + d
    + e + f + g + h}
  \nonumber \\
  & = & i + j
\end{eqnarray}
```

$$
\begin{aligned}
&a+b+c+d+e+f+g+h \\
&\quad = i+j && (3.20)
\end{aligned}
$$

뭐가 문제인지 설명하는 것은 이쯤하고 이제 본격적으로 제대로 된 해결책을 모색해보자.

3.5.2 IEEEeqnarray 환경

`IEEEeqnarray` 환경은 여러 옵션을 갖춘 매우 강력한 명령이다. 여기서는 기본적 기능만을 소개할 것이다. 더 자세한 사항은 사용설명서를 참고하라.[10]

[10] 공식 사용설명서는 CTAN://macros/latex/contrib/IEEEtran/IEEEtran_HOWTO.pdf이다. 부록 F에 `IEEEeqnarray`에 대한 부분이 있다.

IEEEeqnarray 환경을 사용하려면 IEEEtrantools 패키지를 로드하여야 한다. 그리고 문서의 전처리부에 다음과 같이 써넣는다.

```
\usepackage{IEEEtrantools}
```

IEEEeqnarray의 장점은 식의 여러 컬럼에 대하여 각각 정렬 방식을 지정할 수 있다는 것이다. 보통 적용하는 정렬은 {rCL}이다. 즉 세 컬럼에 대하여 첫 컬럼은 왼쪽정렬, 두 번째는 가운데정렬하면서 좌우에 조금 여백을 주고(그래서 소문자 c가 아니라 대문자 C를 쓴 것이다), 세 번째 컬럼은 왼쪽정렬하라는 뜻이다.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c
  \\
  & = & d + e + f + g + h
  + i + j + k \nonumber\\
  && \negmedspace {} + l + m + n + o
  \\
  & = & p + q + r + s
\end{IEEEeqnarray}
```

$$
\begin{IEEEeqnarray}{rCl}
a &=& b + c \\
&=& d + e + f + g + h + i + j + k \\
&&\quad + l + m + n + o \\
&=& p + q + r + s
\end{IEEEeqnarray}
$$

(3.21) (3.22) (3.23)

컬럼 수의 제한은 없다. {c} 하나만을 쓴다면 모든 수식을 가운데정렬한다. {rCll}이라고 하면 왼쪽 정렬되는 네 번째 컬럼을 추가하는 것으로서 주석이나 설명을 위하여 쓸 수 있다. 수식 모드로 식자되는 l, c, r, L, C, R 이외에 s, t, u가 있어 각각 왼쪽, 가운데, 오른쪽에 오는 텍스트 모드로 지정할 수 있다. 추가 여백을 크기순으로 ., /, ? 부호로 나타낸다.[11]

eqnarray를 썼을 때와 비교하여 등호 주변의 공백 크기가 어떻게 달라졌는지 주의깊게 보라.

3.5.3 일반적 사용법

흔하게 만나는 문제를 해결하기 위해 IEEEeqnarray를 사용하는 방법에 대해 알아보자.

수식 번호가 겹쳐나타나는 식 (3.17)의 수식 번호 겹침 문제는

```
\IEEEeqnarraynumspace
```

이 명령으로 해결할 수 있다. 수식 번호 있는 행에 추가하면 수식 번호가 식자되는 크기를 계산하여 필요한 만큼 전체 수식을 왼쪽으로 옮긴다. 옮겨지는 크기는 수식 번호의 크기에 따라 결정된다. 즉 다음과 같은 좋지 않은 결과를

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f + g + h
  + i + j + k  + l\\
  & = & l + m + n
\end{IEEEeqnarray}
```

$$
\begin{IEEEeqnarray}{rCl}
a &=& b + c \\
&=& d + e + f + g + h + i + j + k + l \\
&=& l + m + n
\end{IEEEeqnarray}
$$

(3.24) (3.25) (3.26)

이렇게 수정할 수 있다.

[11] 3.9.1절에 공백 유형에 대한 언급이 더 있다.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f + g + h
  + i + j + k + l
  \IEEEeqnarraynumspace\\
  & = & l + m + n.
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b + c && (3.27)\\
&= d + e + f + g + h + i + j + k + l && (3.28)\\
&= l + m + n. && (3.29)
\end{aligned}
$$

좌변이 너무 길 경우에 좋지 않은 \lefteqn 명령을 쓰지 말고 IEEEeqnarray의 \IEEEeqnarraymulticol 명령을 쓰는 것이 좋다. 모든 상황에서 잘 작동한다.

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d + e + f
    + g + h
  }\nonumber\\ \quad
  & = & i + j
  \\
  & = & k + l + m
\end{IEEEeqnarray}
```

$$
\begin{aligned}
&a + b + c + d + e + f + g + h\\
&\quad = i + j && (3.30)\\
&\quad = k + l + m && (3.31)
\end{aligned}
$$

이 명령은 tubular 환경의 \multicolumns 명령과 완전히 같다. 첫 번째 인자 3은 세 개의 컬럼을 합칠 것이라는 의미이고 l은 합쳐진 컬럼을 왼쪽 정렬한다는 뜻이다.

등호의 위치를 맞추려면 \quad 명령을 넣어서 쉽게 할 수 있다.[12] 예를 들어본다.

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d + e + f
    + g + h
  }\nonumber\\ \qquad\qquad
  & = & i + j
  \\
  & = & k + l + m
\end{IEEEeqnarray}
```

$$
\begin{aligned}
&a + b + c + d + e + f + g + h\\
&\qquad\qquad = i + j && (3.32)\\
&\qquad\qquad = k + l + m && (3.33)
\end{aligned}
$$

식이 둘 이상의 줄로 나누어지는 때 LaTeX은 줄 처음에 나오는 +나 −를 이항연산 부호로 보지 않고 양수 음수 부호처럼 취급한다. 따라서 이 연산부호 앞에 빈 항 {}를 넣어야 한다. 다음은 잘못된 예이다.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c
  \\
  & = & d + e + f + g + h
  + i + j + k \nonumber\\
  && + l + m + n + o
  \\
  & = & p + q + r + s
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b + c && (3.34)\\
&= d + e + f + g + h + i + j + k\\
&\quad + l + m + n + o && (3.35)\\
&= p + q + r + s && (3.36)
\end{aligned}
$$

다음과 같이 해야 올바르다.

[12] quad 한 개가 대부분의 경우에 적절할 거라고 생각한다.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c
  \\
  & = & d + e + f + g + h
  + i + j + k \nonumber\\
  && \negmedspace {} + l + m + n + o
  \\
  & = & p + q + r + s
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b + c && (3.37)\\
&= d + e + f + g + h + i + j + k \\
&\quad + l + m + n + o && (3.38)\\
&= p + q + r + s && (3.39)
\end{aligned}
$$

+과 l 사이의 간격이 달라졌음이 보이는가. 빈 항을 넣어서 {} + l라고 하는 것은 +가 양수 부호가 아니라 항 사이의 덧셈을 나타내는 부호임을 강제하는 방법이다. 또한 {}와 + 사이에 생겨나는 미세한 간격은 \negmidspace (negative medium space) 명령으로 없앴다.

수식 번호를 붙이지 않을 행에는 \nonumber를 적어주면 번호가 붙지 않는다. 만약 이 행에 \label{eq:...}가 정의되어 있다면 이 label은 다음 번 수식 번호를 가리키게 된다. 레이블은 행 나눔 명령 \\ 직전에 두거나 그 레이블을 달 수식 뒤에 붙이도록 하라. 소스 코드를 읽기 쉬워지는 것과 더불어 \IEEEmulticol 명령이 라벨 정의 뒤에 잇달아 나올 때 발생할 수 있는 컴파일 오류를 방지한다.

별표붙은 환경을 사용하면 수식 번호가 아무 데도 붙지 않는다. 이럴 경우에 수식 번호를 나타내어야 할 필요가 있으면 \IEEEyesnumber 명령을 이용한다.

```
\begin{IEEEeqnarray*}{rCl}
  a & = & b + c \\
  & = & d + e \IEEEyesnumber\\
  & = & f + g
\end{IEEEeqnarray*}
```

$$
\begin{aligned}
a &= b + c \\
&= d + e \qquad (3.40)\\
&= f + g
\end{aligned}
$$

\IEEEyessubnumber를 통해 하위 번호도 손쉽게 붙일 수 있다.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c
  \IEEEyessubnumber\\
  & = & d + e
  \nonumber\\
  & = & f + g
  \IEEEyessubnumber
\end{IEEEeqnarray}
```

$$
\begin{aligned}
a &= b + c && (3.40a)\\
&= d + e \\
&= f + g && (3.40b)
\end{aligned}
$$

3.6 배열과 행렬

배열을 식자하려면 array 환경을 쓴다. tabular 환경과 비슷하다. 열 분리에 & 행 분리에 \\ 명령을 사용한다.

```
\begin{equation*}
  \mathbf{X} = \left(
    \begin{array}{ccc}
      x_1 & x_2 & \ldots \\
      x_3 & x_4 & \ldots \\
      \vdots & \vdots & \ddots
    \end{array} \right)
\end{equation*}
```

$$\mathbf{X} = \left(\begin{array}{ccc} x_1 & x_2 & \ldots \\ x_3 & x_4 & \ldots \\ \vdots & \vdots & \ddots \end{array} \right)$$

array 환경은 조각적으로 정의된 함수를 표현하는 데도 사용할 수 있다. 이 때는 오른쪽 닫는 괄호 위치에 \right.를 표시해야 한다.

```
\begin{equation*}
  |x| = \left\{
    \begin{array}{rl}
      -x & \text{if } x < 0,\\
      0 & \text{if } x = 0,\\
      x & \text{if } x > 0.
    \end{array} \right.
\end{equation*}
```

$$|x| = \left\{ \begin{array}{rl} -x & \text{if } x < 0,\\ 0 & \text{if } x = 0,\\ x & \text{if } x > 0. \end{array} \right.$$

amsmath의 cases 환경은 이것을 좀더 간편하게 쓸 수 있게 한다. 다음과 같다.

```
\begin{equation*}
  |x| =
  \begin{cases}
    -x & \text{if } x < 0,\\
    0 & \text{if } x = 0,\\
    x & \text{if } x > 0.
  \end{cases}
\end{equation*}
```

$$|x| = \begin{cases} -x & \text{if } x < 0,\\ 0 & \text{if } x = 0,\\ x & \text{if } x > 0. \end{cases}$$

행렬을 array로 조판할 수 있다. 그러나 amsmath 패키지가 다양한 matrix 환경을 제공하므로 이를 이용하는 것이 더 좋다. 여닫는 부호에 따라 여섯 가지가 있다. matrix(여닫는 부호 없음), pmatrix(둥근 괄호 ()), bmatrix(각진 괄호 []), Bmatrix(활 괄호 {}), vmatrix(세로선 |), Vmatrix(겹세로선 ‖). 이 환경에서는 array에서처럼 열의 수를 미리 지정하지 않아도 된다. 최대 10개까지인데 조절할 수 있다. (그런데 열이 10개나 되는 행렬을 표기할 일이 흔할까?)

```
\begin{equation*}
  \begin{matrix}
    1 & 2 \\
    3 & 4
  \end{matrix} \qquad
  \begin{bmatrix}
    p_{11} & p_{12} & \ldots
    & p_{1n} \\
    p_{21} & p_{22} & \ldots
    & p_{2n} \\
    \vdots & \vdots & \ddots
    & \vdots \\
    p_{m1} & p_{m2} & \ldots
    & p_{mn}
  \end{bmatrix}
\end{equation*}
```

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \qquad \begin{bmatrix} p_{11} & p_{12} & \ldots & p_{1n} \\ p_{21} & p_{22} & \ldots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \ldots & p_{mn} \end{bmatrix}$$

3.7 수식 모드에서의 간격

LaTeX이 조판한 수식 내의 간격이 만족스럽지 못하다면 특별한 간격 명령으로 그것을 조절할 수 있다. 간격 조절 명령으로는 `\,` 명령은 $\frac{3}{18}$ quad (␣), `\:`는 $\frac{4}{18}$ quad (␣), 그리고 `\;`는 $\frac{5}{18}$ quad (␣)에 해당하는 간격을 넣는다. 백슬래시를 붙인 스페이스 `\␣`는 일반적인 단어 사이 띄어쓰기 간격에 해당하는 중간 크기의 간격을, `\quad` (␣)와 `\qquad` (␣␣)는 큰 간격을 넣는다. `\quad`의 크기는 사용중인 폰트의 M자 폭(1em)에 해당한다. `\!`는 반대 방향으로 $-\frac{3}{18}$ quad 만큼 (−␣)을 이동한다.

```
\begin{equation*}
  \int_1^2 \ln x \mathrm{d}x
  \qquad
  \int_1^2 \ln x \,\mathrm{d}x
\end{equation*}
```

$$\int_1^2 \ln x\mathrm{d}x \qquad \int_1^2 \ln x\,\mathrm{d}x$$

미분 기호의 'd'를 그대로 이탤릭으로 표현하기도 하지만 로만 정자로 인쇄하는 관행도 있다. 그럴 때에는 `\ud` 명령(upright d)을 "`$\,\mathrm{d}$`"로 정의해두고 쓸 수 있다. (d 앞에 ␣ 만큼의 간격을 준 것에 주의하여 보라. 매번 간격 명령을 주지 않아도 되게 되었다.) 이 명령을 정의하는 `\newcommand`를 전처리부에 넣어둔다.

```
\newcommand{\ud}{\,\mathrm{d}}

\begin{equation*}
 \int_a^b f(x)\ud x
\end{equation*}
```

$$\int_a^b f(x)\,\mathrm{d}x$$

중적분 등 적분기호를 이어 조판해야 할 때에 적분기호 사이가 너무 벌어지는 것을 볼 수 있다. `\!`로 간격을 줄일 수 있지만 amsmath는 더 쉬운 방법을 제공한다. `\iint`, `\iiint`, `\iiiint`, `\idotsint` 명령이 그것이다.

```
\newcommand{\ud}{\,\mathrm{d}}

\begin{IEEEeqnarray*}{c}
  \int\int f(x)g(y)
                \ud x \ud y \\
  \int\!\!\!\int
        f(x)g(y) \ud x \ud y \\
  \iint f(x)g(y)  \ud x \ud y
\end{IEEEeqnarray*}
```

$$\begin{aligned}
\int\int f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y \\
\iint f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y \\
\iint f(x)g(y)\,\mathrm{d}x\,\mathrm{d}y
\end{aligned}$$

더 상세한 내용을 보려면 $\mathcal{AMS}$-LaTeX과 함께 배포되는 전자문서 `testmath.tex`이나 *The LaTeX Companion* [3]의 제8장을 참고하라.

3.7.1 허깨비 글자

`\phantom` 명령을 쓰면 자리는 차지하면서 출력물에는 나오지 않도록 글자 만큼의 간격을 얻을 수 있다. `^`나 `_`의 수직 위치 정렬을 LaTeX이 너무 심하게 잘 해주기 때문에 원하지 않는 결과를 보일 때 이를 이용하여 수정할 수 있다. 실례를 보는 것이 제일 좋을 것이다.

```
\begin{equation*}
{}^{14}_{6}\text{C}
\qquad \text{versus} \qquad
{}^{14}_{\phantom{1}6}\text{C}
\end{equation*}
```

$$ {}^{14}_{6}\text{C} \qquad \text{versus} \qquad {}^{14}_{6}\text{C} $$

화학의 반응식과 동위원소를 대거 조판해야 한다면 mhchem 패키지가 도움이 된다.

3.8 수학 폰트에 대하여

여러 가지 수학 폰트들이 63페이지의 표 3.14에 정리되어 있다.

```
$\Re \qquad
 \mathcal{R} \qquad
 \mathfrak{R} \qquad
 \mathbb{R} \qquad $
```

$$ \Re \qquad \mathcal{R} \qquad \mathfrak{R} \qquad \mathbb{R} \qquad $$

뒤의 두 가지는 amssymb나 amsfonts 패키지가 필요하다.

가끔 적당한 폰트 크기를 설정해야 할 때가 있다. 다음과 같은 네 가지 명령으로 크기를 조절한다.

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

분수의 분모나 분자 위치에 $\sum$이 오면 별다른 지정이 없는 한 텍스트 스타일로 식자된다. 이것을 디스플레이 스타일로 하라고 알려주려면 다음과 같이 한다.

```
\begin{equation*}
 P = \frac{\displaystyle{
   \sum_{i=1}^n (x_i- x)
   (y_i- y)}}
   {\displaystyle{\left[
   \sum_{i=1}^n(x_i-x)^2
   \sum_{i=1}^n(y_i- y)^2
   \right]^{1/2}}}
\end{equation*}
```

$$ P = \frac{\displaystyle{ \sum_{i=1}^n (x_i- x) (y_i- y)}} {\displaystyle{\left[\sum_{i=1}^n(x_i-x)^2 \sum_{i=1}^n(y_i- y)^2 \right]^{1/2}}} $$

위의 예와 같이 수식의 스타일을 바꾸거나 강제하면 큰 연산자 부호와 리미트가 표시되는 방식이 달라진다.

3.8.1 굵은 부호

LaTeX에서 굵은 기호를 얻기는 상당히 어렵다. 아마도 의도적인 것은 아닌가 싶은데 많은 비전문가들이 이것을 남용하는 경향이 있기 때문일 것이다. \mathbhf를 주면 글자(letter)에 대해서는 볼드체를 얻을 수 있지만 글자가 로만체(곧게 선 서체)로 되어버린다. 수학에 쓰이는 글자들은 이탤릭이기 때문에 생각과 다르게 나오는 것일 수 있다. 게다가 그리스어 소문자는 아예 굵은 글자 자체가 나오지 않는다. \boldmath라는 명령이 있어서 기호문자에 대해서도 동작하기는 하지만 수식 모드를 빠져나가야지만 쓸 수 있다.

```
$\mu, M \qquad
\mathbf{\mu}, \mathbf{M}$
\qquad \boldmath{$\mu, M$}
```

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

amsbsy 패키지(amsmath의 일부이다)와 tools 묶음의 bm 패키지에 있는 \boldsymbol 명령은 이것을 훨씬 쉽게 할 수 있게 해준다.

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

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

3.9 수학적 문단(정리, 보조정리 등)

수학적인 글쓰기에 있어서 "보조정리(Lemma)", "정의(Definition)", "공리(Axiom)"와 같은 문단을 식자해야 할 때가 있다.

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

name 인자는 해당 "정리"에 고유하게 부여한 짧은 키워드이다. *text* 인자에는 그 실제 인쇄될 "정리"의 이름을 써준다.

대괄호 안에 넣어주는 선택 인자는 생략가능하다. 두 가지가 모두 "정리"의 번호에 사용될 것을 가리킨다. *counter* 인자는 미리 선언한 어떤 "theorem"의 *name*을 지정한다. 그러면 새로운 "theorem"은 여기 적힌 이름을 가진 theorem과 동일한 번호 순차를 공유할 것이다. *section* 인자는 "theorem"의 번호가 section 단위에 연계되도록 설정하는 것이다.

\newtheorem을 문서의 전처리부에 실행한 후 본문에서 다음과 같은 명령을 사용할 수 있다.

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

amsthm 패키지는 다음과 같은 명령을 사용할 수 있게 한다. \theoremstyle{*style*} 명령으로 모든 theorem은 사전에 정의된 세 가지 양식 가운데 하나를 선택하여 정의하도록 되어 있다. definition (두꺼운 제목, 로만체 본문), plain (두꺼운 제목, 이탤릭체 본문), remark (이탤릭체 제목, 로만체 본문).

이것이면 충분하다. 다음 예제를 보면 모든 것이 명확해질 것이며 \newtheorem의 뭔가 이해하기 어려운 점을 분명하게 알게 될 것이다.

우선 theorem을 정의한다.

```
\theoremstyle{definition} \newtheorem{law}{Law}
\theoremstyle{plain}      \newtheorem{jury}[law]{Jury}
\theoremstyle{remark}     \newtheorem*{margt}{Margaret}
```

```
\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{jury}
You will disregard the last
statement.\end{jury}
\begin{margt}No, No, No\end{margt}
\begin{margt}Denis!\end{margt}
```

Law 1. Don't hide in the witness box
Jury 2 (The Twelve)**.** *It could be you! So beware and see law 1.*
Jury 3. *You will disregard the last statement.*
Margaret. No, No, No
Margaret. Denis!

"Jury" 정리는 "Law" 정리와 동일한 카운터를 이용하도록 되어 있다. 그래서 이전의 "Laws" 정리에 이어서 번호가 붙었다. 대괄호 안에 놓인 인자는 그 정리의 제목 또는 표제를 설정한다.

```
\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 3.9.1. 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" 정리는 현재 section의 번호에 연계되어 있다. chapter나 subsection 같은 다른 단위를 지정하는 것도 가능하다.

만약 정리 문단의 모양 글꼴 배치 등을 아주 세밀하게 조절하고 싶다면 ntheorem 패키지가 지나치게 많은 옵션을 제공하고 있다.

3.9.1 증명, QED 부호

amsthm 패키지는 `proof` 환경을 제공한다.

```
\begin{proof}
 Trivial, use
 \begin{equation*}
   E=mc^2.
 \end{equation*}
\end{proof}
```

Proof. Trivial, use

$$E = mc^2.$$

□

"증명종료" 부호가 외따로 한 줄을 차지하는 상황을 피하도록 하기 위해 `\qedhere` 명령을 쓸 수 있다.

```
\begin{proof}
 Trivial, use
 \begin{equation*}
   E=mc^2. \qedhere
 \end{equation*}
\end{proof}
```

Proof. Trivial, use

$$E = mc^2.$$ □

안타깝게도 이러한 수정이 `IEEEeqnarray`에서는 제대로 동작하지 않는다.

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray*}{rCl}
    a & = & b + c \\
    & = & d + e. \qedhere
  \end{IEEEeqnarray*}
\end{proof}
```

Proof. This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \end{aligned}$$ □

그 이유는 IEEEeqnarray가 항상 array의 양측에 신축가능한 공간만을 가지는 눈에 보이지 않는 컬럼을 두기 때문이다. 그래서 IEEEeqnarray에서는 컬럼 안의 식이 수평으로 가운데 오도록 작동한다. \qedhere 명령은 사실 이 눈에 보이지 않는 공간을 벗어나서 놓여야만 한다. 그러나 이 공간이 사용자 눈에 보이지 않기 때문에 이와 같은 일이 일어나게 된다.

이 문제를 수정하기 위해 신축 공백을 명시적으로 정의하는 간단한 방법이 있다.

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray*}{+rCl+x*}
    a & = & b + c \\
    & = & d + e. & \qedhere
  \end{IEEEeqnarray*}
\end{proof}
```

Proof. This is a proof that ends with an equation array:

$$\begin{aligned} a &= b + c \\ &= d + e. \end{aligned}$$ □

{+rCl+x*}에서 +는 신축 공백을 의미한다. 하나는 식의 왼쪽에(이것은 별도로 정의하지 않으면 IEEEeqnarray가 자동으로 설정한다) 그리고 다른 하나는 오른쪽에 온다. 그러나 이제 오른쪽에는 신축 공백 칼럼 **뒤에** x라는 빈 컬럼을 추가하고 있다. 이 컬럼은 마지막 줄의 \qedhere 명령이 놓일 자리를 마련하기 위한 목적으로만 쓰이는 것이다. 마지막의 *는 IEEEeqnarray에게 원치 않는 + 공간을 추가하지 말라고 알려주는 것이다.

수식 번호의 경우에도 비슷한 문제가 발생한다. 다음 두 가지 사례를 비교해보라.

```
\begin{proof}
  This is a proof that ends
  with a numbered equation:
  \begin{equation}
    a = b + c.
  \end{equation}
\end{proof}
```

Proof. This is a proof that ends with a numbered equation:

$$a = b + c. \tag{3.41}$$

□

```
\begin{proof}
  This is a proof that ends
  with a numbered equation:
  \begin{equation}
    a = b + c. \qedhere
  \end{equation}
\end{proof}
```

Proof. This is a proof that ends with a numbered equation:

$$a = b + c. \tag{3.42}$$

□

두 번째 예제에서 □가 첫 번째 것보다 수식에 훨씬 가깝게 놓였다는 것을 알 수 있다.

이와 유사하게 IEEEeqnarray의 끝에 QED 부호를 두는 올바른 방법은 다음과 같다.

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray}{+rCl+x*}
    a & = & b + c \\
    & = & d + e. \\
    &&& \qedhere\nonumber
  \end{IEEEeqnarray}
\end{proof}
```

Proof. This is a proof that ends with an equation array:

$$a = b + c \tag{3.43}$$
$$= d + e. \tag{3.44}$$

□

이것을 아래와 비교해보라.

```
\begin{proof}
  This is a proof that ends
  with an equation array:
  \begin{IEEEeqnarray}{rCl}
    a & = & b + c \\
    & = & d + e.
  \end{IEEEeqnarray}
\end{proof}
```

Proof. This is a proof that ends with an equation array:

$$a = b + c \tag{3.45}$$
$$= d + e. \tag{3.46}$$

□

3.10 수학 기호 목록

다음 수학기호 목록은 **수식모드**(*math mode*)에서 사용가능한 기호문자를 열거한 것이다.

아래 기호 중의 일부는 amssymb 패키지를 로드하여야 사용할 수 있다.[13] $\mathcal{AMS}$ 패키지와 그 폰트에 관해서는 CTAN://pkg/amslatex을 보라. 완전한 기호문자 일람표를 보려면 CTAN://info/symbols/comprehensive를 참조하라.

표 3.1: 수식모드의 액센트

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

표 3.2: 그리스 문자

대문자 Alpha를 위한 `\Alpha`나 `\Beta`라는 명령은 없다. 왜냐하면 이 글자들은 로마자 공통이고 `A`, `B`와 같이 입력하여 A, B를 얻을 수 있기 때문이다.

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

[13] 이 표는 David Carlisle이 처음 작성하고 그 후 Josef Ikadlec이 확장하여 수정한 `symbols.tex`에서 가져온 것이다.

표 3.3: 이항 관계 연산자

다음 연산자 부호 앞에 \not를 붙이면 부정 기호를 얻을 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]	$\sqsupset$	\sqsupset [a]	$\bowtie$	\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 기호는 latexsym 패키지를 요구한다.

표 3.4: 이항 연산자

$+$	+	$-$	-		
$\pm$	\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]	$\rhd$	\rhd [a]	$\ddagger$	\ddagger
$\unlhd$ [a]	\unlhd [a]	$\unrhd$	\unrhd [a]	$\wr$	\wr

표 3.5: 큰 연산자

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

표 3.6: 화살표

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

[a] 해당 기호는 latexsym 패키지가 필요함

표 3.7: 문자의 위아래로 오는 화살표

$\overrightarrow{AB}$	\overrightarrow{AB}	$\underrightarrow{AB}$	\underrightarrow{AB}
$\overleftarrow{AB}$	\overleftarrow{AB}	$\underleftarrow{AB}$	\underleftarrow{AB}
$\overleftrightarrow{AB}$	\overleftrightarrow{AB}	$\underleftrightarrow{AB}$	\underleftrightarrow{AB}

표 3.8: 여닫는 부호

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

표 3.9: 큰 여닫는 부호

$\left(\right.$	\lgroup	$\left.\right)$	\rgroup	$\left\lmoustache\right.$	\lmoustache
$\left\lvert\right.$	\arrowvert	$\left\rVert\right.$	\Arrowvert	$\left\lvert\right.$	\bracevert
$\left.\right\rmoustache$	\rmoustache				

표 3.10: 기타 부호

…	\dots	⋯	\cdots	⋮	\vdots	⋱	\ddots
ℏ	\hbar	𝚤	\imath	𝚥	\jmath	ℓ	\ell
ℜ	\Re	ℑ	\Im	ℵ	\aleph	℘	\wp
∀	\forall	∃	\exists	℧	\mho [a]	∂	\partial
′	'	′	\prime	∅	\emptyset	∞	\infty
∇	\nabla	△	\triangle	□	\Box [a]	◇	\Diamond [a]
⊥	\bot	⊤	\top	∠	\angle	√	\surd
♢	\diamondsuit	♡	\heartsuit	♣	\clubsuit	♠	\spadesuit
¬	\neg or \lnot	♭	\flat	♮	\natural	♯	\sharp

[a] 해당 기호는 latexsym 패키지가 필요함

표 3.11: 수학 기호가 아닌 것

다음 기호들은 텍스트 모드에서도 사용할 수 있다.

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

표 3.12: $\mathcal{AMS}$: 여닫는 부호

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

표 3.13: $\mathcal{AMS}$: 그리스와 히브리 문자

ϝ	\digamma	ϰ	\varkappa	ב	\beth	ג	\gimel	ד	\daleth

표 3.14: 수학 알파벳

그밖의 수학 폰트에 관해서는 100페이지의 표 6.4를 보라.

예문	명령	필요패키지
ABCDEabcde1234	\mathrm{ABCDE abcde 1234}	
ABCDEabcde1234	\mathit{ABCDE abcde 1234}	
*ABCDEabcde*1234	\mathnormal{ABCDE abcde 1234}	
$\mathcal{ABCDE}$	\mathcal{ABCDE abcde 1234}	
$\mathscr{ABCDE}$	\mathscr{ABCDE abcde 1234}	mathrsfs
𝔄𝔅ℭ𝔇𝔈abcde1234	\mathfrak{ABCDE abcde 1234}	amsfonts or amssymb
𝔸𝔹ℂ𝔻𝔼∋⊮⊭⊮⊉	\mathbb{ABCDE abcde 1234}	amsfonts or amssymb

표 3.15: $\mathcal{AMS}$: 이항 관계 연산자

∔	\dotplus	∙	\centerdot		
⋉	\ltimes	⋊	\rtimes	⋇	\divideontimes
⋓	\doublecup	⋒	\doublecap	⧵	\smallsetminus
⩒	\veebar	⩛	\barwedge	⩚	\doublebarwedge
⊞	\boxplus	⊟	\boxminus	⊝	\circleddash
⊠	\boxtimes	⊡	\boxdot	⊚	\circledcirc
⊺	\intercal	⊛	\circledast	⋌	\rightthreetimes
⋎	\curlyvee	⋏	\curlywedge	⋋	\leftthreetimes

표 3.16: $\mathcal{AMS}$: 이항 연산자

⋖	\lessdot	⋗	\gtrdot	≒	\doteqdot
⩽	\leqslant	⩾	\geqslant	≓	\risingdotseq
⪅	\eqslantless	⪆	\eqslantgtr	≒	\fallingdotseq
≦	\leqq	≧	\geqq	≖	\eqcirc
⋘	\lll or \llless	⋙	\ggg	≗	\circeq
≲	\lesssim	≳	\gtrsim	≜	\triangleq
⪅	\lessapprox	⪆	\gtrapprox	≏	\bumpeq
≶	\lessgtr	≷	\gtrless	≎	\Bumpeq
⋚	\lesseqgtr	⋛	\gtreqless	∼	\thicksim
≨	\lesseqqgtr	≩	\gtreqqless	≈	\thickapprox
≼	\preccurlyeq	≽	\succcurlyeq	≊	\approxeq
⋞	\curlyeqprec	⋟	\curlyeqsucc	∽	\backsim
≾	\precsim	≿	\succsim	⋍	\backsimeq
⪷	\precapprox	⪸	\succapprox	⊨	\vDash
⫅	\subseteqq	⫆	\supseteqq	⊩	\Vdash
∥	\shortparallel	⫐	\Supset	⊪	\Vvdash
◀	\blacktriangleleft	⊐	\sqsupset	϶	\backepsilon
▷	\vartriangleright	∵	\because	∝	\varpropto
▶	\blacktriangleright	⋐	\Subset	≬	\between
⊵	\trianglerighteq	⌢	\smallfrown	⋔	\pitchfork
◁	\vartriangleleft	∣	\shortmid	⌣	\smallsmile
⊴	\trianglelefteq	∴	\therefore	⊏	\sqsubset

표 3.17: $\mathcal{AMS}$: 화살표

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

표 3.18: $\mathcal{AMS}$: 이항 연산자와 화살표의 부정

≮	\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 3.19: $\mathcal{AMS}$: 기타

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

제 4 장

특별한 기능

큰 규모의 문서를 작성할 때 색인 생성, 문헌 목록 관리와 같은 특별한 기능이 필요하다. LaTeX은 이런 일을 할 수 있도록 해준다. 훨씬 많은 특별한 기능과 확장된 기능이 있으며 *LaTeX Manual* [1]과 *The LaTeX Companion* [3]에 상세히 설명되어 있다.

4.1 문헌 목록

문헌 목록은 `thebibliography` 환경으로 만든다. 각 문헌항목을

`\bibitem[`*label*`]{`*marker*`}`

명령으로 시작하여 입력한다. *marker*는 문서 내에서 이 도서나 논문 등의 문헌을 인용하는데 다음과 같이 사용한다.

`\cite{`*marker*`}`

선택 인자 *label*을 입력하지 않으면 각 항목은 일련번호가 자동으로 붙게 된다.

숫자 레이블에 얼마 만큼의 공간이 필요한지를 `\begin{thebibliography}` 명령 뒤에 적어넣는데 다음 페이지의 예제에서 99라고 했으므로 항목 수가 99를 넘지 않을 것이라고 예상할 수 있다.

큰 문서 작업을 하고 있다면 BibTeX 프로그램의 사용을 고려해볼 수 있다. BibTeX은 대부분의 TeX 배포판에 포함되어 있다. 이 프로그램이 하는 일은 문헌 목록 데이터베이스를 유지하면서 현재 쓰고 있는 문서에서 인용된 것만을 추려내어 참고문헌 목록으로 만들어준다. BibTeX에 의하여 생성되는 참고문헌 목록은 자신이 만들고자 하는 문헌목록과 인용방식에 의한 스타일시트에 입각하여 모양이 갖추어진다. 여러 학술지나 단행본에서 채택하고 있는 문헌목록과 인용방식의 스타일이 제공된다.

```
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)

4.2 색인

책에 있는 색인(찾아보기)은 매우 유용하다. LaTeX에서는 makeindex 프로그램의 도움을 받아 상당히 손쉽게 색인을 만들 수 있다. 기본적인 색인 생성 명령에 대해서만 소개하겠다. 더 깊은 내용을 알고 싶다면 *The LaTeX Companion* [3]을 참고하라.

색인 생성을 가능하게 하려면 makeidx 패키지를 전처리부에 선언하여야 한다.

```
\usepackage{makeidx}
```

그리고 색인 생성을 시작하라는 의미의 특별한 색인 명령을 준다.

```
\makeindex
```

이 명령은 전처리부에 놓인다.

색인으로 들어갈 내용은 다음과 같이 지정한다.

```
\index{key@formatted_entry}
```

*formatted_entry*는 실제 색인에 인쇄되어 나타날 표제어의 모양이고 *key*는 정렬을 위해 사용되는 문자열이다. *formatted_entry*는 생략할 수 있으며 이 때는 *key*가 사용된다. 색인 목록에서 가리킬 텍스트상의 위치에 index 명령을 적어넣는다. 표 4.1이 이 명령의 사용법을 보여주고 있다.

입력 파일을 LaTeX이 처리하는 과정에서 \index 명령을 만나면 색인에 들어갈 항목표제와 현재 페이지의 번호를 특별한 파일에 기록해둔다. 입력 파일과 이름이 같고 확장자가 .idx인 파일이 생성된다. makeindex 프로그램을 .idx 파일에 대하여 실행할 수 있다.

```
makeindex filename
```

makeindex 프로그램은 색인을 정렬하여 같은 이름에 확장자가 .ind인 파일을 만든다. LaTeX이 다시 입력 파일을 처리할 때 색인 인쇄 명령을 만나면 이 정렬된 색인을 문서

표 4.1: \index 명령의 사용법

Example	Index Entry	Comment
\index{hello}	hello, 1	Plain entry
\index{hello!Peter}	Peter, 3	Subentry under 'hello'
\index{Sam@\textsl{Sam}}	*Sam*, 2	Formatted entry
\index{Lin@\textbf{Lin}}	**Lin**, 7	Formatted entry
\index{Kaese@K\"ase}	**Käse**, 33	Formatted entry
\index{ecole@\'ecole}	école, 4	Formatted entry
\index{Jenny|textbf}	Jenny, **3**	Formatted page number
\index{Joe|textit}	Joe, *5*	Formatted page number

안으로 불러들인다. 색인 인쇄 명령은 다음과 같다.

`\printindex`

showidx 패키지는 모든 색인 항목을 문서의 왼쪽 여백에 표시해준다. 문서를 교열할 때 색인 점검에 유용하다.

\index 명령은 주의깊게 사용하지 않으면 문서의 레이아웃에 영향을 미칠 수 있다는 사실을 알아두자. \index 명령 주변에 의도하지 않은 공백이 들어가지 않도록 하는 것이 중요하다.

```
My Word \index{Word}. As opposed
to Word\index{Word}. Note the
position of the full stop.
```

My Word . As opposed to Word. Note the position of the full stop.

makeindex는 아스키 범위를 넘어서는 문자를 처리하지 못한다. 그러므로 정렬이 제대로 되게 하려면 앞서 보인 예에서 Käse나 école을 처리할 때처럼 @로 정렬 위치를 잘 알려주는 것이 필요하다.

[역자의 보충] 한국어 문서의 색인 작성을 위해서 makeindex를 그대로 쓰면 만족스러운 결과를 얻기 어렵다. ko.TEX은 이를 위하여 komkindex라는 프로그램을 제공한다. ko.TEX과 함께 배포되므로 이미 설치되어 있을 것이다. 이 프로그램은 한 가지 유용한 옵션을 가지고 있는데, 라틴 문자보다 한글이 색인에서 먼저 나타나게 하고 싶으면 -k 옵션을 줄 수 있다는 것이다. 이밖에 최근 유니코드 문자를 처리할 수 있는 색인 유틸리티가 발전하는 중이다. xindy, xindex 등이 그 예이다.

4.3 면주 장식

문서의 면주[1]를 사용자가 만들 수 있도록 하는 명령을 제공하는 패키지가 있다. Piet van Oostrum이 작성한 fancyhdr이다.[2] 영문판 lshort 문서의 페이지 스타일이 이 패키지를

[1] [역주] "면주(面柱)"란 책의 본문 외곽에 편장절의 표제 등을 적어넣은 것을 가리키는 조판 용어이다. *header*와 *footer*를 이용하여 만드는 *page style*을 가리키는 의미로 사용하였다. 편의상 header를 상단면주, footer를 하단면주라고 하였다.

[2] CTAN://macros/latex/contrib/supported/fancyhdr.

이용하여 만들어진 것이다.[3]

```
\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 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} % 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

상하단 면주를 입맛대로 고치려 할 때 까다로운 점은 현재 장표제나 절표제를[4] 거기에 넣는 것이다. LaTeX에서 이 작업은 두 단계로 이루어진다. 상하단의 면주 정의에는 `\rightmark`와 `\leftmark` 명령을 이용하여 현재의 절표제와 장표제를 각각 가리키게 해 둔다. 그리고 장이 바뀌거나 새로운 절이 시작될 때마다(즉 `\chapter` 명령이나 `\section` 명령이 불릴 때마다) 이 명령에 할당된 값을 새로운 것으로 대체하게 하는 것이다.

유연성을 확보하고 충돌을 방지하기 위해서 `\chapter`나 `\section` 명령이 `\rightmark`와 `\leftmark` 명령을 직접 수정하지는 않는다. 그 대신 `\chaptermark`, `\sectionmark`, `\subsectionmark` 같은 다른 명령을 호출하는데 이 명령들이 `\rightmark`와 `\leftmark`를 정의하는 역할을 맡고 있다.

면주의 장표제 모양을 바꾸고 싶으면 `\chaptermark`를 재정의(renewcommand)하면 된다.[5]

그림 4.1에 영문판 lshort의 면주 모양과 똑같이 페이지 스타일을 정의한 예를 보였다. 아무튼 이 패키지의 사용설명서를 읽어보기를 권한다.

[3] [역주] 한국어판은 영문판 페이지 스타일을 비슷하게 흉내내었지만 조금 다른 방법으로 만들었다.

[4] [역주] "running heading(s)"이라고 한다.

[5] [역주] 예시된 코드를 잘 보면 `\markboth`와 `\markright`라는 명령을 볼 수 있다. `\rightmark`와 `\markright`를 혼동하기 쉬운데 `\rightmark`는 러닝 헤딩을 가지고 있는 문자열 매크로이고 `\markright`는 오른쪽(홀수쪽) 러닝 헤딩을 부여하는 명령이라고 기억하면 된다. 이 책자의 왼쪽 상단 면주에는 장의 제목이, 오른쪽 상단 면주에는 절의 제목이 들어가 있다.

4.4 Verbatim 패키지

verbatim 환경에 대해서는 이 책의 앞부분에서 이미 보았다. 여기서는 verbatim 패키지에 대해 배워보려 한다. verbatim 패키지는 기본적으로 원래의 verbatim 환경이 가지고 있는 한계를 보충하기 위해서 verbatim 환경을 재구현한 것이다. 그 자체로 엄청난 것은 아니지만 새로운 기능을 추가한 것이 있고 그 때문에 여기서 이 패키지를 다루는 것이다. verbatim 패키지는 다음과 같은 명령을 제공한다.

`\verbatiminput{`*filename*`}`

이 명령은 외부 텍스트 파일의 내용을 verbatim 환경 안에 넣으면서 불러들이도록 한다.

이 패키지에 대해 더 알고 싶으면 사용설명서 [10]을 읽어보라.

4.5 패키지의 추가 설치

대부분의 LaTeX 설치배포판은 엄청나게 많은 스타일 패키지를 미리 설치해준다. 그렇지만 인터넷에서 구할 수 있는 패키지도 많다. 스타일 패키지에 대하여 우선 방문해보아야 할 곳은 CTAN (http://www.ctan.org)이다.

geometry, hyphenat 등 많은 패키지들은 두 개의 파일로 배포된다. 하나는 .ins 파일이고 다른 하나는 .dtx 파일이다. readme.txt가 있어서 패키지에 대해 간략히 설명하고 있을 수 있다. 당연히 이 파일을 제일 먼저 읽어야 한다.

어떤 경우든 패키지 파일을 내려받았다면 이것을 다음과 같이 처리하여야 한다. (a) 자신의 TeX 시스템에게 스타일 파일에 대해 알려주는 것. (b) 패키지 문서를 얻는 것.

먼저 (a)에 해당하는 작업을 해보자.

1. LaTeX을 .ins 파일에 대하여 실행한다. 그렇게 하면 .sty 파일이 풀려나온다.
2. .sty 파일을 TeX 시스템의 적절한 위치로 옮긴다. 일반적으로 .../*localtexmf*/tex/latex이라는 폴더 아래에 설치한다.
3. TeX 시스템의 파일네임 데이터베이스를 갱신한다. 배포판에 따라서 이것을 하는 명령이 조금씩 다르다. TeX Live는 texhash 또는 mktexlsr, web2c는 mktexlsr, MiKTeX은 initexmf --update-fndb 등이며 편리하게 쓰는 GUI를 이용할 수도 있다.

그 다음으로 문서를 얻는 방법을 알아본다.

1. XeLaTeX을 .dtx 파일에 대하여 실행한다. 그러면 .pdf 파일이 만들어진다. 교차참조가 바르게 나타나게 하려면 XeLaTeX을 몇 번 실행해야 할 것이다.
2. XeLaTeX의 실행 과정에서 .idx 파일이 생성되었는지 확인한다. 만약 이 파일이 생겨나지 않았으면 설명 문서에 색인이 없는 것이므로 5단계로 진행한다.
3. 색인을 생성해야 한다면 다음 명령을 실행한다.
   `makeindex -s gind.ist` *name*
   (여기서 *name*은 파일 이름을 확장자 없이 써넣는다는 뜻이다.)

4. XeLaTeX을 `.dtx`에 대해 한 번 더 실행한다.

5. 마지막으로 만들어진 `.pdf` 문서를 즐겁게 읽어보면 된다.

가끔 `.glo` (glossary) 파일이 생겨나 있을 때가 있다. 이것은 용어집을 만들기 위한 파일인데 다음 명령을 4와 5단계 사이에 실행하여야 한다.

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

5단계로 넘어가기 전에 XeLaTeX을 마지막에 한 번 실행해야 한다는 것을 기억하자.

4.6 LaTeX과 PDF

글쓴이: Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

PDF (Portable Document Format)는 흔하게 볼 수 있는 강력한 하이퍼텍스트 문서 형식이다. 하이퍼텍스트란 웹페이지에서 볼 수 있는 바와 같이 특정 어구에 하이퍼링크를 달아서 동일 문서 내 또는 다른 문서로 클릭하여 이동하거나 참조할 수 있게 하는 장치이다. LaTeX으로 말하자면 `\ref`와 `\pageref`, `\cite`가 모두 하이퍼링크로 이루어지게 할 수 있으며 목차나 색인 등도 하이퍼링크를 갖춘 구조로서 조판되게 할 수 있다.

오늘날 많은 웹페이지가 HTML (Hypertext Markup Language)로 작성된다. 이 형식은 과학 문헌을 작성하는 경우에 중요한 결점 두 가지가 있다.

1. HTML 자체만으로는 수학식을 포함하기 어렵다. 이를 위한 표준이 있기는 하지만 지원하는 브라우저가 많지 않거나 필요한 폰트가 없을 때가 많다.[6]

2. HTML 문서를 인쇄할 수 있기는 하나 결과는 플랫폼과 브라우저에 따라 천차만별이다. 그 결과물은 LaTeX으로 만든 문서와 비교할 때 품질 면에서 너무나 현격한 차이가 있다.

LaTeX을 HTML으로 변환하는 다양한 시도가 있었다. 그 가운데 일부는 표준 LaTeX 입력 파일로부터 읽을 만한 웹 페이지를 만들어낼 수 있다는 의미에서 그럭저럭 성공적인 것도 있다. 그러나 이렇게 하기 위해서는 너무나 많은 것을 희생해야 한다. 다양한 패키지를 활용한 복잡한 LaTeX 문서가 되면 당장 제대로 되지 않는 것들이 나타나기 시작한다. 자신이 작성한 문서의 타이포그래피적 품위를 유지하면서도 웹을 통하여 출판하고자 한다면 PDF를 고려해보는 것이 좋다. 이 포맷은 문서의 레이아웃을 그대로 유지하면서 하이퍼텍스트를 포함하기 때문이다. 현대의 브라우저들은 PDF 문서를 브라우저 내에서 바로 보여주는 부가기능을 대부분 가지고 있다.

현재의 모든 TeX 엔진은 PDF를 즉시 생성할 수 있다. 이 입문서를 잘 따라왔다면 PDF 만들기도 이미 익숙해진 방법으로 할 수 있다.

4.6.1 하이퍼텍스트 링크

hyperref 패키지는 LaTeX PDF 파일에 멋진 기능을 추가해준다.

[6] [역주] 여기서 '표준'이라 한 것은 아마 MathML을 지칭하는 듯하다. 한편 요즘 웹상에서 수학식을 표현하는 방법이 개발되어 이전의 어려움을 대부분 극복한 것은 사실이다. 특히 MathJax가 유명하다. https://www.mathjax.org/. MathJax는 LaTeX 형식으로 입력되는 수식을 잘 표현한다.

1. 용지 크기는 문서 클래스에 선언한 것에 맞추어진다.
2. 문서상의 모든 참조는 하이퍼링크로 만들어진다.

`\usepackage{hyperref}` 문장을 전처리부에 **마지막** 문장으로 적어넣는다.
hyperref 패키지의 동작을 지정하기 위해 여러 가지 옵션을 사용할 수 있다.

- `\usepackage[`*options*`]{hyperref}`와 같이 옵션을 쉼표로 분리하여 나열하거나
- `\hypersetup{`*options*`}`라는 별도의 명령으로 옵션을 설정할 수 있다.

사용가능한 옵션을 몇 가지 보이겠다. 로만 폰트로 적힌 것이 기본값(default)이다.

`bookmarks (=true, false)` 문서의 pdf 책갈피(bookmarks)를 보이거나 보이지 않게 한다.

`unicode (=false,true)` Acrobat의 책갈피에 비라틴문자를 허용할 것인지의 여부이다.

`pdftoolbar (=true,false)` Acrobat의 툴바를 보이거나 보이지 않게 한다.

`pdfmenubar (=true,false)` Acrobat 메뉴를 보이거나 숨긴다.

`pdffitwindow (=false,true)` PDF를 열었을 때 전체 창에 맞추어 확대할 것인지 여부이다.

`pdftitle (={text})` Acrobat의 **문서 정보**에 표시될 문서 제목을 정한다.

`pdfauthor (={text})` **문서 정보**에 나타날 저자명을 정한다.

`pdfnewwindow (=false,true)` 문서의 링크를 눌러서 다른 문서로 이동할 때 새 창을 열 것인가의 여부.

`colorlinks (=false,true)` 하이퍼링크가 걸린 텍스트를 색상 프레임으로 둘러쌀 것인지(`false`) 아니면 색을 입힐 것인지(`true`)를 나타낸다. 각 링크 종류에 따라 색상이 다르게 나타나는데 해당 옵션은 다음과 같다(표시된 색상은 기본값이다).

`linkcolor (=red)` 내부 링크 (section, 페이지 등)

`citecolor (=green)` 인용 링크 (참고문헌)

`filecolor (=magenta)` 파일로의 링크

`urlcolor (=cyan)` URL 링크 (메일, 웹)

기본값으로 충분하다면

```
\usepackage{hyperref}
```

책갈피를 열고 링크를 색상 텍스트로 나타내려면 (`=true`는 생략가능)

```
\usepackage[bookmarks,colorlinks]{hyperref}
```

위와 같이 설정한다.

PDF를 인쇄용으로 만드는 경우에 색상 링크는 좋은 방법이 아니다. 흑백 프린터로 인쇄하면 회색으로 나오기 때문에 읽기 어려워진다. 반면 프레임은 화면으로만 보이고 인쇄되지 않는다.

```
\usepackage{hyperref}
\hypersetup{colorlinks=false}
```

또는 링크 색상을 모두 black으로 하려면

```
\usepackage{hyperref}
\hypersetup{colorlinks,%
            citecolor=black,%
            filecolor=black,%
            linkcolor=black,%
            urlcolor=black,%
            pdftex}
```

PDF 파일의 문서 정보에 들어갈 내용을 다음과 같이 설정한다.

```
\usepackage[pdfauthor={Pierre Desproges},%
            pdftitle={Des femmes qui tombent},%
            pdftex]{hyperref}
```

자동 하이퍼링크 말고, 특정의 링크를 임의로 주기 위해서는 다음 명령을 쓴다.

`\href{`*url*`}{`*text*`}`

아래와 같이 하면

```
\href{http://www.ctan.org}{CTAN} 웹사이트.
```

이 코드는 "CTAN"이라는 링크를 만들어낸다. "CTAN" 부분을 누르면 CTAN 웹사이트로 이동할 것이다.

링크가 가리키는 목적지가 URL이 아니라 파일인 경우에 `\href` 명령에 'http://' 부분을 제외하고 쓴다.

```
전체 문서는 \href{manual.pdf}{여기}를 보시오.
```

이렇게 하면 "전체 문서는 여기를 보시오"라는 텍스트가 생성된다. "여기"라고 되어 있는 부분을 클릭하면 (`manual.pdf`라는 파일이 읽고 있는 문서와 같은 폴더에 있을 때) 파일이 열릴 것이다.

전자우편 주소에 하이퍼링크를 걸어두면 문서를 읽던 독자가 즉시 이메일을 보낼 수 있다.

```
\author{Mary Oetiker $<$\href{mailto:mary@oetiker.ch}%
        {mary@oetiker.ch}$>$
```

위의 예에서 저자의 이름에 이메일 주소 하이퍼링크를 걸지 않고 이메일 주소를 쓰면서 거기에 다시 링크를 걸었는데 이렇게 한 이유는 저자 이름에만 이메일 주소 링크를

`\href{mailto:mary@oetiker.ch}{Mary Oetiker}`

이런 식으로 걸어둔 경우에 PDF를 화면으로 읽고 있다면 문제없이 동작하지만 인쇄했을 때 이메일 주소가 사라져서 보이지 않게 되기 때문이다.

4.6.2 링크 관련 문제

pdfTeX의 경고 메시지로 이런 것이 있다.

```
! pdfTeX warning (ext4): destination with the same
  identifier (name{page.1}) has been already used,
  duplicate ignored
```

이것은 카운터가 재설정되었을 때 나타난다. 예를 들면 book 클래스의 \mainmatter 명령을 주면 페이지 번호 카운터를 1로 재설정하여 제1장의 페이지 번호가 1이 되는데 "서문"의 페이지 번호도 인쇄시에는 i로 나타나 있어서 구별되지만 내부적으로는 똑같이 1이기 때문에 "1페이지"를 가리키는 링크는 더이상 한 군데를 가리키지 않아서 혼선을 빚게 된다. 그래서 경고 메시지에서 duplicate가 ignore되었다고 하는 것이다.

plainpages=false 옵션을 두면 카운터 재설정 문제는 해결되는데 문제는 이것이 오직 페이지 카운터에만 해당한다는 점이다. 극단적으로 hypertexnames=false로 해버리는 방법이 있지만 이렇게 하면 색인에서 페이지 링크가 더이상 먹지 않는다.

4.6.3 북마크 관련 문제

pdf 북마크는 편리한 장치지만 거기에 나타나는 모양이 항상 예상대로인 것이 아니다. 왜냐하면 북마크는 "단순히 텍스트"일 뿐이라서 보통 LaTeX으로 표현되는 텍스트 가운데 사용할 수 없는 문자가 많다. hyperref은 이런 문제를 만나면 그저 무시하면서 경고를 보여준다.

```
Package hyperref Warning:
Token not allowed in a PDFDocEncoded string:
```

이 문제를 피해가려면 북마크용 텍스트를 별도로 제공해주면 된다.

\texorpdfstring{*TeX text*}{*Bookmark Text*}

수학식이 북마크에 들어가는 경우에 흔히 발생하는 일이다.

```
\section{\texorpdfstring{$E=mc^2$}%
        {E = mc ** 2}}
```

$E = mc^2$을 절표제로 사용하면서 북마크에는 "E = mc ** 2"을 넣으라고 지시하는 것이다.

유니코드 문서를 작성하고 있다면 hyperref 패키지에 unicode 옵션을 부여하면 북마크에 유니코드 문자를 사용할 수 있다. \texorpdfstring하는 경우에 허용할 수 있는 문자 범위가 넓어진다.

4.7 XeLaTeX과 PDF

글쓴이: Axel Kielhorn <A.Kielhorn@web.de>

앞 절에서 논의한 내용 대부분이 XeLaTeX에도 적용된다.

XeTeX과 XeLaTeX 관련 정보가 http://wiki.xelatex.org/docu.php의 위키 사이트에 있다.

4.7.1 폰트

전통(레거시) 텍 엔진의 `tfm` 기반 폰트 이외에, XeLaTeX은 운영체제에 등록된 어떤 폰트라도 사용할 수 있다. `Linux Libertine` 폰트가 설치되어 있다면 간단히

```
\usepackage{fontspec}
\setmainfont[Ligatures=TeX]{Linux Libertine}
```

이와 같이 전처리부에 선언하는 것으로 문서에 이 폰트를 사용할 수 있다. 같은 글꼴 가족에 해당하는 이탤릭 글꼴과 볼드 글꼴을 대개 찾아내기 때문에 `\textit`이나 `\textbf` 명령도 잘 작동한다. 예전에 가상 폰트나 별도의 폰트를 통해서 하던 폰트 변형, 확장, 대체 기능이 오늘날은 오픈타입(OpenType) 기술에 의한 font feature로 구현되어 있는데 이를 활성화할 수도 있다. 문자 집합이 확장되어서 하나의 폰트가 라틴 문자, 그리스 문자, 키릴 문자와 대응 합자(ligature)들을 모두 갖추고 있게 되었다.

최소 두 종류의 숫자 모양을 갖춘 폰트가 많다. old style 숫자라 불리는 것과 라이닝 숫자라 불리는 것인데, old style 숫자는 베이스라인 아래로 내려가는 모양으로 되어 있다. 그리고 가변폭(proportional) 숫자("1"과 "0"이 서로 다른 너비를 가진다)와 고정폭(monospaced) 숫자를 가지고 있을 수도 있는데 고정폭은 표 작성에 알맞다.

```
\newfontfamily\LLln[Numbers=Lining]{(font)}
\newfontfamily\LLos[Numbers=OldStyle]{(font)}
\newfontfamily\LLlnm[Numbers=Lining,Numbers=Monospaced]{(font)}
\newfontfamily\LLosm[Numbers=OldStyle,Numbers=Monospaced]{(font)}
```

대부분의 오픈타입 폰트에 표준 합자(fl fi ffi)가 들어 있는데 st, ct, tz 같은 희귀한 합자나 역사적 합자를 가진 것도 있다. 이런 것을 과학적 문서에 쓸 수는 없지만 소설 등이라면 괜찮을 것이다. 합자 사용을 지정하는 방법을 다음 예제에서 나타내었다.

```
\setmainfont[Ligatures=Rare]{(font)}
\setmainfont[Ligatures=Historic]{(font)}
\setmainfont[Ligatures=Historic,Ligatures=Rare]{(font)}
```

모든 폰트가 합자들을 다 갖추고 있는 것은 아니므로 폰트 견본을 보거나 직접 시험해 보라. 어떤 합자는 언어의존적이다. 예컨대 폴란드어의 (fk) 합자는 영어에서는 사용되지 않는 것이다. 폴란드어 특유의 합자를 활성화하려면 다음과 같이 한다.

```
\setmainfont[Language=Polish]{(font)}
```

어떤 폰트(예컨대 상업용 폰트인 Adobe Garamond Premier Pro)에 포함된 대체 글리프를 TeX Live 2010 이후 버전에서 XeLaTeX이 기본적으로 활성화한다.[7] 그 결과 특유의 "Q" 문자 꼬리가 그 다음에 오는 "u" 아래까지 닿는 모양으로 인쇄된다. 이 기능을 억제하려면

```
\setmainfont[Contextuals=NoAlternate]{(font)}
```

이와 같이 지정한다.

XeLaTeX의 폰트 사용법에 대해 더 자세한 사항은 fontspec 패키지 설명서를 읽어보면 된다.

[7] 그 이전 버전에서는 되지 않았으나 이 버전 이후 변경된 사항이다.

오픈타입 폰트를 얻을 수 있는 곳

TeX Live를 설치하여 사용하고 있다면 .../texmf-dist/fonts/opentype 폴더를 열어 보면 상당히 많은 오픈타입 폰트가 있음을 알 수 있다. LaTeX 작업을 위해서는 이대로 사용가능하며 시스템 전체적으로 활용하고 싶다면 운영체제에 설치하여 써도 된다.

각 폰트는 한 번만 설치한다. 그렇지 않으면 이상한 결과를 얻을 수 있다.

자신의 컴퓨터에는 있는 폰트라도 다른 사람은 그 폰트를 가지고 있지 않을 수 있다는 사실을 기억하자. Zapfino 폰트는 fontspec 설명서에도 나와 있는 것인데 Mac OS X에 포함되어 있지만 윈도우즈 컴퓨터에는 없을 수 있다.[8]

유니코드 입력

폰트에 포함되는 글자들은 많아졌는데 키보드의 키 숫자는 늘어나지 않았다. 아스키 문자 이외의 글자들을 어떻게 입력할까?

특정 외국어의 입력 분량이 상당하다면 그 언어에 맞는 키보드를 설치하여 키 배열을 인쇄해서 보면서 입력할 수 있겠다. (대부분의 운영체제에는 가상 키보드 기능이 있으므로 그것을 잘 활용해보라.)

입력할 문자가 몇 개 안 되는 경우라면 문자표에서 선택하여 붙여넣는 방법이 있다.

X 윈도우 시스템처럼 비아스키 문자 입력에 여러 방법을 제공하는 환경도 있으며 vim이나 Emacs와 같이 특수 문자 입력 방법을 제공하는 에디터도 있다. 사용할 수 있는 도구의 설명서를 읽어보도록 하자.

4.7.2 한국어 폰트

글쓴이: 한국어판을 위하여 추가

한국어 문서를 작성하기 위해서는 적어도 세 종류의 폰트가 필요하다. 하나는 라틴 문자를 위한 것이고 다른 하나는 한글 폰트이며 마지막으로 한자를 식자할 폰트가 있어야 한다. 한글과 한자를 같은 폰트로 식자할 수 있다면(즉 폰트에 한글과 한자가 모두 갖추어져 있다면) 라틴 문자용 폰트와 한글/한자용 폰트 두 종류가 있으면 된다. 조금 복잡한 상태인 까닭은 근본적으로 한글 한자 등 우리 문서 작성에 필요한 모든 자면(글리프)을 다 갖춘 신뢰할 만한 폰트가 존재하지 않기 때문이다.

XeLaTeX 지원 koTeX은 폰트에 대하여 다른 설정이 없으면 라틴 문자에 대해서는 Latin Modern 폰트를, 한글과 한자에 대해서는 나눔글꼴을 기본 폰트로 적용한다.

koTeX의 XeLaTeX 지원 패키지는 fontspec을 기본적으로 로드하여 활성화하기 때문에, 라틴 문자 영역에 대한 것은 앞선 절에서 소개한 내용을 그대로 적용할 수 있다.

한글 폰트는 main/sans/mono 폰트(각각 rm, sf, tt에 해당)를 다음과 같은 명령으로 설정한다. 이 명령은 전처리부의 `\usepackage{kotex}` 이후에 두면 된다.

`\setmainhangulfont{`*fontname*`}[`*options*`]`

*fontname*은 해당 폰트의 이름이다.

*options*를 활용하는 한 예시는 다음과 같다.

[8] Zapfino Extra라는 상업용 폰트가 있다.

```
\setmainhangulfont{UnBatang}[BoldFont=UnBatang-Bold,AutoFakeSlant]
```

여기서 BoldFont는 \bfseries로 식자할 폰트 이름을 적어주는 것이고[9] AutoFakeSlant는 이탤릭체를 써야 할 상황에서 기울어진 글꼴을 사용하라는 의미이다. 우사체(fakeslant) 사용은 권장하지 않는 바이지만 구태여 하려 한다면 이렇게 할 수 있다는 것이다.

폰트 설정 방법과 옵션에 관하여서는 xetexko 매뉴얼[25]을 꼭 참조해 보아야 한다.

한글 오픈타입 또는 트루타입 폰트는 대부분 XeLaTeX에서 사용가능하다. 손쉽게 사용할 수 있는 한글 폰트는 은 글꼴, 나눔글꼴, 함초롬LVT 폰트, Noto CJK Fonts 등이 있다. 은 글꼴 트루타입은 TeX Live로 배포되므로 별도로 설치할 필요가 없지만 다른 글꼴은 스스로 운영체제에 설치해야 사용가능하다.

한글 폰트의 이름을 알아내기 위해서는 otfinfo라는 프로그램(TeX Live에 포함되어 있는 유틸리티 프로그램)을 사용하는 것이 좋다. 예를 들어 나눔명조 폰트 파일 C:\Windows\Fonts\NanumMyeongjo.ttf에 대하여 이 명령을

```
otfinfo -i "C:\Windows\Fonts\NanumMyeongjo.ttf"
```

적용하면 "Full Name"을 알려준다. 되도록 여기에 표시되는 이름을 사용하여야 한다.

필요하다면 라틴 문자와 문장부호에 대해서 한글 폰트를 함께 적용하는 방법도 있다. 가장 쉬운 방법은 라틴 문자를 설정하는 \setmainfont 명령에도 한글 폰트 이름을 똑같이 지정하는 것이다.

4.7.3 XeLaTeX과 pdfLaTeX의 호환성

XeLaTeX과 pdfLaTeX 사이에 몇 가지 차이점이 있다.

- XeLaTeX 문서는 유니코드(UTF-8)로 입력한다. 반면 pdfLaTeX은 입력 인코딩이 다양해서 주의를 기울여야 한다.[10]
- microtype 패키지가 XeLaTeX에는 아직 적용되지 않는다. 그렇지만 pdfTeX의 미세타이포그래피 기능 중의 여백돌출(character protrusion)에 대한 지원은 이미 개발이 시작되었다.
- 폰트에 관련된 기능은 완전히 새로워졌다. (Latin Modern 폰트로 만족한다면 큰 차이를 느끼지 못하겠지만.)

4.8 발표자료 만들기

글쓴이: Daniel Flipo <Daniel.Flipo@univ-lille1.fr>

자신의 과학적 연구성과를 발표할 때, 칠판을 이용하거나 투명 필름 슬라이드를 이용하거나 아니면 노트북 컴퓨터에서 프레젠테이션 소프트웨어를 사용하여 직접 시현하거나 한다.

[9] [역주] UnBatang의 경우에는 BoldFont를 별달리 지정하지 않아도 찾아낼 것이지만 어떤 폰트들은 BoldFont를 명시해야 할 때가 있다.

[10] [역주] pdfLaTeX도 utf-8 입력을 (어떻게든) 처리할 수는 있다. 그러나 pdfLaTeX은 엔진 자체가 유니코드를 이해하고 처리한다고 말할 수는 없다는 차이점이 있다. koTeX의 pdfLaTeX 대응 버전은 utf-8 입력을 기본으로 하고 있었다.

beamer 클래스를 이용하면 한때 잘 쓰던 PowerPoint나 LibreOffice로 만든 것 같은 멋진 발표자료를 PDF로 만들 수 있다. PDF는 리더만 있으면 어떤 컴퓨터에서나 볼 수 있으므로 편리성이 훨씬 높다.

beamer 클래스는 graphicx, color, hyperref 패키지의 옵션을 활용하여 화면 프레젠테이션에 적합한 결과를 만들어낸다.

```
\documentclass[10pt]{beamer}
\mode<beamer>{%
  \usetheme[hideothersubsections,
            right,width=22mm]{Goettingen}
}

\title{Simple Presentation}
\author[D. Flipo]{Daniel Flipo}
\institute{U.S.T.L. \& GUTenberg}
\titlegraphic{\includegraphics[width=20mm]{USTL}}
\date{2005}

\begin{document}

\begin{frame}<handout:0>
  \titlepage
\end{frame}

\section{An Example}

\begin{frame}
  \frametitle{Things to do on a Sunday Afternoon}
  \begin{block}{One could \ldots}
    \begin{itemize}
      \item walk the dog\dots \pause
      \item read a book\pause
      \item confuse a cat\pause
    \end{itemize}
  \end{block}
  and many other things
\end{frame}
\end{document}
```

그림 4.2: beamer 클래스 샘플 코드

그림 4.2에 있는 코드를 pdfLaTeX이나 XeLaTeX으로 컴파일하면 PDF 파일을 얻는다. 첫 페이지는 표제면이고 두 번째 페이지는 항목들이 발표를 진행함에 따라 단계별로 표시되는 것을 보여준다.

beamer 클래스의 장점 중 하나는 다른 조작 없이 PDF를 바로 얻을 수 있다는 것이다. prosper는 PostScript 단계를 거쳐야 했고 ppower4는 추가적인 후처리가 필요했다.

beamer 클래스를 쓰면 동일한 입력 파일에서 서로 다른 버전의 ('모드(mode)'에 따라

달라지는) 결과물을 얻어낼 수 있다. 각 모드에 특정한 지시사항은 홑화살괄호(<>) 안에 넣어서 전달한다. 다음과 같은 모드가 있다.

beamer 화면용 발표자료. 앞서 언급한 것이다.

trans 투명 필름 작성용.

handout 인쇄용 문서.

기본 모드는 beamer이다. 전역 옵션으로 다른 모드를 지정한다. `\documentclass[10pt,handout]{beamer}`라고 하는 것은 핸드아웃용으로 조판하라는 것이다.

발표자료는 테마에 따라 그 모양이 달라진다. beamer 클래스가 함께 제공하는 테마 중에서 하나를 골라도 되고 직접 작성할 수도 있다. 더 자세한 정보는 사용설명서 beameruserguide.pdf를 보라.

그림 4.2에 보인 코드를 자세히 살펴보자.

화면용 프레젠테이션에 해당하는 사항으로 `\mode<beamer>`에서 *Goettingen* 테마를 선택하였다. 이 테마는 목차로부터 만들어진 내비게이션 패널을 보여준다. 지시된 옵션 사항은 내비게이션 패널의 사이즈(이 경우 22mm), 위치(본문의 오른쪽)를 설정한다. *hideothersubsections*라는 옵션은 chapter와 현재 소절(subsection)의 타이틀만을 표시하게 하는 것이다. `\mode<trans>`와 `\mode<handout>`에 대해서 특별한 설정이 주어지지 않았기 때문에 이 모드에는 표준 레이아웃이 적용된다.

`\title{}`, `\author{}`, `\institute{}`, `\titlegraphic{}` 명령은 표제지의 내용을 설정하는 것이다. `\title[]{}`과 `\author[]{}`의 선택 인자는 *Goettingen* 테마의 패널에 표시될 제목과 저자명을 지정해준다.

패널에 나타나는 표제와 부속표제는 일반적인 `\section{}`과 `\subsection{}` 명령으로 만든다. 이 명령은 frame 환경의 바깥에 둔다.

화면 바닥에 있는 작은 내비게이션 아이콘으로 문서를 이동할 수 있다. 이것은 테마와는 무관하다.

각 슬라이드 즉 한 화면은 frame 환경 안에 입력한다. 홑화살괄호로 주는 옵션 인자는 발표자료의 버전에 따라 특정 프레임을 생략하려 할 때 쓸 수 있다. 이 예제에서 첫 페이지가 핸드아웃에 나타나지 않도록 `<handout:0>`을 설정했다.

발표 자료 전체 제목과 무관하게 슬라이드 제목을 다는 것이 필요하다. `\frametitle{}` 명령이 이 일을 한다. 부제목이 필요하다면 예제에 있는 대로 block 명령을 사용한다. `\section`과 `\subsection`같은 장절명령은 슬라이드 제목으로 나타나지 않는다.

`\pause` 명령을 쓰면 항목들이 하나씩 차례로 나타나게 할 수 있다. `\only`, `\uncover`, `\alt`, `\temporal` 같은 명령을 활용하여 프레젠테이션에 효과를 줄 수 있으므로 사용법을 확인해보라. 프레젠테이션에 원하는 효과를 주기 위해서 홑화살괄호를 사용하는 경우가 많다.

beamer로 무엇을 할 수 있는지 알기 위해서 사용설명서를 반드시 읽어보도록 하라. 이 패키지는 활발하게 개발이 진행 중이므로 최신 정보를 얻기 위해 웹사이트를 방문하는 것도 좋다. (http://latex-beamer.sourceforge.net/)

제 5 장

수학적 그래프 그리기

대부분의 사람들은 텍스트를 조판하기 위해 LaTeX을 사용한다. 한편, 구조 중심적인 방법으로 저작하는 것은 상당히 편리하기에 LaTeX은, 다소 제한적이기는 하나, 텍스트로부터 시각적인 결과물인 그래프를 그릴 수 있는 몇 가지 수단을 제공한다. 더 나아가 이런 제약들을 해결하기 위한 상당수의 LaTeX 확장판들이 만들어졌다. 이 장에서는 이 중 일부를 배우기로 한다.

5.1 개요

LaTeX을 이용하여 시각적 결과물을 만들어내는 일은 꽤 오래된 전통이다. 우선은 캔버스에 지정된 요소들을 교묘히 짜넣는 것으로 그래프를 그려낼 수 있는 `picture` 환경으로 시작했다. *LaTeX Manual* [1]에서 이에 대한 종합적인 설명을 찾아볼 수 있다. LaTeX 2_ε의 `picture` 환경에서 `\qbezier` 명령을 쓸 수 있는데, "`q`"는 "quadratic"을 의미한다. 수학적인 소양을 어느 정도 요구할 수 있지만 원, 타원, 혹은 현수선 등 자주 쓰이는 곡선들의 경우 이차 베지어 곡선을 통해 만족스러운 근사치를 얻을 수 있다. 이에 더하여 만약 프로그래밍 언어를 이용해 LaTeX 입력 파일에서 `\qbezier` 블록들을 그려낼 수 있다면 `picture` 환경은 강력한 도구가 될 수 있다.

LaTeX에서 직접 그림들을 프로그래밍하는 것은 상당히 제한적이고 피곤한 작업이지만, 여전히 그리 할 유인은 있다. 이런 식으로 작성된 문서는 용량의 측면에서 상대적으로 "작으며," 별도의 그림 파일을 질질 끌고 다닐 필요가 없다는 이점이 있다.

그림 그리기에서 LaTeX이 매우 제한적인 기능만을 제공했던 이러한 상황은 몇 년 전 beamer로 명성이 자자했던 Till Tantau가 Portable Graphics Format pgf와 함께 TikZ (tikz) 패키지를 세상에 내놓으며 완전히 달라졌다. 이 시스템은 완전한 pdf 지원을 포함하여 사용자가 현존하는 모든 TeX 시스템에서 고품질의 벡터 그래프들을 그리는 것을 가능하게 했다.

이것을 기반으로 하여 각기 특정한 목적 가진 수많은 패키지들이 탄생했다. *The LaTeX Graphics Companion* [4]에는 이런 류의 다양한 패키지들이 상세하게 기술되어 있다.

LaTeX관련 그래픽 툴 중에서 가장 앞서가는 것은 아마도 METAPOST일 것이다. 이것은 Donald E. Knuth의 METAFONT에 기반을 두고 있는 독립적인 응용프로그램이다. METAPOST는 METAFONT의 강력하면서도 수학적으로 섬세한 프로그래밍 언어를 사용하지만 METAFONT와는 다르게 캡슐화된(encapsulated) PostScript 파일을 생성해내며 그 코드는 LaTeX, 심지어는 pdfLaTeX 문서에 포함될 수 있다. 이에 대한 소개는 *A User's Manual for METAPOST* [14]혹은 [16]의 튜토리얼에서 볼 수 있다.

LaTeX과 TeX에서의 그래프(그리고 폰트)에 대해 사용자가 사용할 수 있는 전략들을 심도 있게 다루는 토론은 *TeX Unbound* [15]에서 찾아 볼 수 있다.

5.2 picture 환경

글쓴이: Urs Oswald <osurs@bluewin.ch>

앞서 언급했던 picture 환경은 표준 LaTeX의 일부이며 간단한 작업을 할 때나 사용자가 특정 요소를 페이지의 특정한 위치에 정밀하게 배치하고 싶을 때 유용하게 사용될 수 있다. 하지만 이 이상의 심도 있는 그래프 작업을 위해서는 91페이지의 5.3절에 소개되는 TikZ를 찾아보는 편이 좋다.[1]

5.2.1 기초 명령들

picture 환경[2]은 아래의 둘 중 하나의 방법으로 사용한다.

```
\begin{picture}(x,y)...\end{picture}
```

또는

```
\begin{picture}(x,y)(x0,y0)...\end{picture}
```

숫자 x, y, x_0, y_0는 \unitlength를 참조한다. 이는 아래의 명령과 같은 방법으로 언제든 재지정이 가능하지만 picture 환경 내부에서는 조정이 불가능하다.

```
\setlength{\unitlength}{1.2cm}
```

\unitlength의 기본값은 1pt이다. 첫 순서쌍인 (x,y)는 문서 내부에 그림이 들어갈 사각형의 공간을 지정한다. 두번째 쌍(x_0,y_0)은 옵션으로, 지정한 공간의 왼쪽 하단에 임의의 좌표를 배정한다.

[1] [역주] 이는 권장이라기보다 필수 사항이라고 보는 것이 옳다. LaTeX의 전통적인 picture 환경은 너무나 제약이 많고 그 결과물도 아름답지 않기 때문에 좋은 대안이 나와 있는 현재에는 사용할 필요가 없다. 따라서 이 절은 전통을 살펴보는 정도에 의미를 두고, 실제로 그림을 그린다면 본문의 제안대로 TikZ 또는 다른 그림 그리기 패키지를 배울 필요가 있다.

[2] 믿거나 말거나, picture 환경은, 표준 LaTeX 2_ε에서 다른 패키지를 불러들일 필요없이 바로 작동한다.

대부분의 드로잉 명령어는 다음 둘 중 하나의 형태를 띤다.

`\put(`x, y`){`$object$`}`

또는

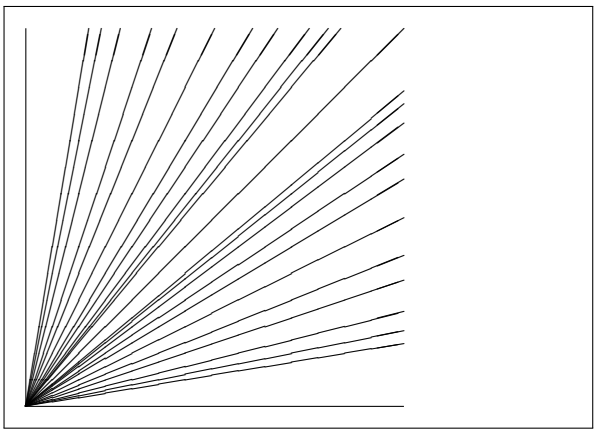
`\multiput(`x, y`)(`$\Delta x, \Delta y$`){`n`}{`$object$`}`

베지어 곡선은 예외이다. 베지어 곡선은 다음의 명령어로 그린다.

`\qbezier(`x_1, y_1`)(`x_2, y_2`)(`x_3, y_3`)`

5.2.2 선분

```
\setlength{\unitlength}{5cm}
\begin{picture}(1,1)
  \put(0,0){\line(0,1){1}}
  \put(0,0){\line(1,0){1}}
  \put(0,0){\line(1,1){1}}
  \put(0,0){\line(1,2){.5}}
  \put(0,0){\line(1,3){.3333}}
  \put(0,0){\line(1,4){.25}}
  \put(0,0){\line(1,5){.2}}
  \put(0,0){\line(1,6){.1667}}
  \put(0,0){\line(2,1){1}}
  \put(0,0){\line(2,3){.6667}}
  \put(0,0){\line(2,5){.4}}
  \put(0,0){\line(3,1){1}}
  \put(0,0){\line(3,2){1}}
  \put(0,0){\line(3,4){.75}}
  \put(0,0){\line(3,5){.6}}
  \put(0,0){\line(4,1){1}}
  \put(0,0){\line(4,3){1}}
  \put(0,0){\line(4,5){.8}}
  \put(0,0){\line(5,1){1}}
  \put(0,0){\line(5,2){1}}
  \put(0,0){\line(5,3){1}}
  \put(0,0){\line(5,4){1}}
  \put(0,0){\line(5,6){.8333}}
  \put(0,0){\line(6,1){1}}
  \put(0,0){\line(6,5){1}}
\end{picture}
```



선분은 다음의 명령을 사용하여 그린다.

`\put(`x, y`){\line(`x_1, y_1`){`$length$`}}`

명령 `\line`은 두 인자를 가진다:

1. 방향 벡터 (direction vector),
2. 길이 (length).

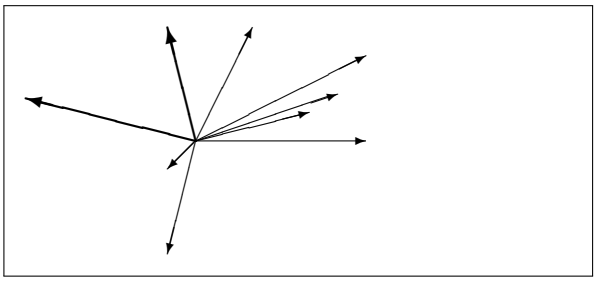
방향 벡터의 성분은 다음 정수로만 제한된다.

$$-6,\ -5,\ \ldots,\ 5,\ 6,$$

그리고 두 성분은 서로소(1 이외의 공약수가 없는 수들)이어야 한다. 제시된 그림은 제1 사분면에서 가능한 25가지의 모든 기울기값을 보여준다.[3] 길이(length) 인자는 \unitlength에 비례한다. 이는 수직선의 경우에는 수직 좌표, 나머지 경우에는 수평 좌표를 의미한다.

5.2.3 화살표

```
\setlength{\unitlength}{0.75mm}
\begin{picture}(60,40)
  \put(30,20){\vector(1,0){30}}
  \put(30,20){\vector(4,1){20}}
  \put(30,20){\vector(3,1){25}}
  \put(30,20){\vector(2,1){30}}
  \put(30,20){\vector(1,2){10}}
  \thicklines
  \put(30,20){\vector(-4,1){30}}
  \put(30,20){\vector(-1,4){5}}
  \thinlines
  \put(30,20){\vector(-1,-1){5}}
  \put(30,20){\vector(-1,-4){5}}
\end{picture}
```



화살표는 다음 명령을 사용하여 그린다.

\put(x,y){\vector(x_1,y_1){*length*}}

화살표의 경우, 방향 벡터의 성분이 선분의 경우에 비해 더욱 제한되어 다음의 정수만을 사용할 수 있다.

$$-4,\ -3,\ \ldots,\ 3,\ 4.$$

이 성분들 또한 서로소(1 이외의 공약수가 없는 수들)이어야 한다. 좌상향으로 뻗은 두 화살표에서 \thicklines 명령의 효과를 볼 수 있다.

5.2.4 원

다음 명령은 중심이 (x, y)이며 (반지름이 아니라) 지름이 *diameter*인 원을 그린다.

\put(x,y){\circle{*diameter*}}

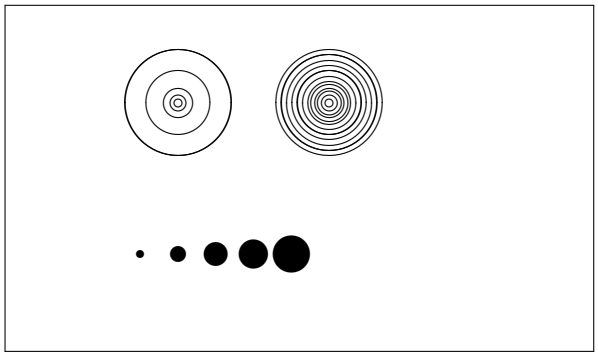
picture 환경은 지름이 14 mm 정도까지만 허용하며, 이 값 이하라고 해서 모든 수치를 허용하지도 않는다. \circle* 명령은 disk(내부가 차있는 원)를 만든다.

[3] [역주] 앞선 역자주에서 picture 환경을 그저 살펴보는 용도로만 사용하라고 권고하였지만 선분의 기울기 제약은 일찍부터 문제가 되었던 터라 이를 해결한 pict2e 패키지가 있음을 지적해둔다. 이 패키지를 이용하면 선분의 기울기 문제를 비롯한 많은 제약을 해결할 수 있다.

```
\setlength{\unitlength}{1mm}
\begin{picture}(60, 40)
  \put(20,30){\circle{1}}
  \put(20,30){\circle{2}}
  \put(20,30){\circle{4}}
  \put(20,30){\circle{8}}
  \put(20,30){\circle{16}}
  \put(20,30){\circle{32}}

  \put(40,30){\circle{1}}
  \put(40,30){\circle{2}}
  \put(40,30){\circle{3}}
  \put(40,30){\circle{4}}
  \put(40,30){\circle{5}}
  \put(40,30){\circle{6}}
  \put(40,30){\circle{7}}
  \put(40,30){\circle{8}}
  \put(40,30){\circle{9}}
  \put(40,30){\circle{10}}
  \put(40,30){\circle{11}}
  \put(40,30){\circle{12}}
  \put(40,30){\circle{13}}
  \put(40,30){\circle{14}}

  \put(15,10){\circle*{1}}
  \put(20,10){\circle*{2}}
  \put(25,10){\circle*{3}}
  \put(30,10){\circle*{4}}
  \put(35,10){\circle*{5}}
\end{picture}
```

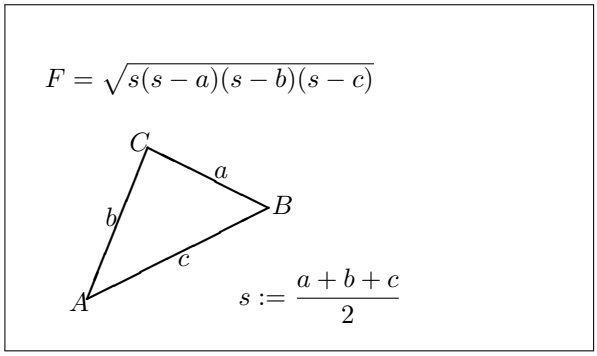


선분의 경우와 마찬가지로 사용자는 eepic이나 pstricks같은 패키지로 눈을 돌려야 하는 상황도 있을 수 있다. 이 패키지들에 대한 자세한 설명은 *The LaTeX Graphics Companion* [4]를 보라.

picture 환경내에서 자체적으로 해결하는 방법도 있다. 사용자가 필요한 계산(이를 프로그램에 맡길 수도 있지만)을 하는 데 불편함이 없다면 베지어 곡선을 패치하여 임의의 원과 타원을 그려내는 것이 가능하다. *Graphics in LaTeX 2ε* [16]에 예제와 Java 소스 파일들이 있다.

5.2.5 텍스트와 수식

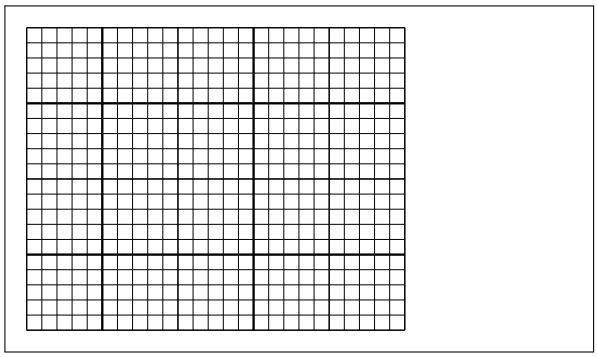
```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,5)
  \thicklines
  \put(1,0.5){\line(2,1){3}}
  \put(4,2){\line(-2,1){2}}
  \put(2,3){\line(-2,-5){1}}
  \put(0.7,0.3){$A$}
  \put(4.05,1.9){$B$}
  \put(1.7,2.95){$C$}
  \put(3.1,2.5){$a$}
  \put(1.3,1.7){$b$}
  \put(2.5,1.05){$c$}
  \put(0.3,4){$F=
    \sqrt{s(s-a)(s-b)(s-c)}$}
  \put(3.5,0.4){$\displaystyle
    s:=\frac{a+b+c}{2}$}
\end{picture}
```



이 예가 보여주듯 텍스트와 수식은 평범하게 \put 명령을 사용하여 picture 환경에 써넣을 수 있다.

5.2.6 \multiput과 \linethickness

```
\setlength{\unitlength}{2mm}
\begin{picture}(30,20)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){26}%
    {\line(0,1){20}}
  \multiput(0,0)(0,1){21}%
    {\line(1,0){25}}
  \linethickness{0.15mm}
  \multiput(0,0)(5,0){6}%
    {\line(0,1){20}}
  \multiput(0,0)(0,5){5}%
    {\line(1,0){25}}
  \linethickness{0.3mm}
  \multiput(5,0)(10,0){2}%
    {\line(0,1){20}}
  \multiput(0,5)(0,10){2}%
    {\line(1,0){25}}
\end{picture}
```



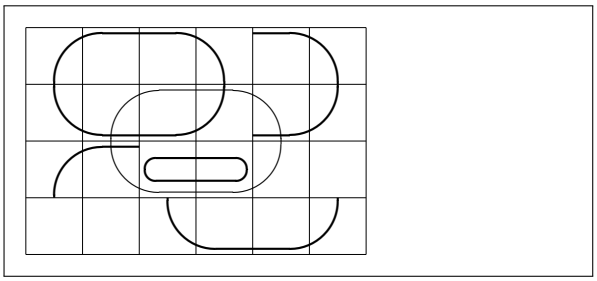
\mulitput 명령은 네개의 인자를 갖는다.

\multiput$(x,y)(\Delta x,\Delta y)\{n\}\{object\}$

네 인자는 시작점, 그래픽 객체가 다음 위치로 가는 이동 벡터(translation vector), 객체의 수, 그리고자 하는 객체(object)이다. \linethickness 명령은 수직선이나 수평선에 적용되지만 경사 선분과 원에는 적용되지 않는다. 그런 한편 이차 베지어 곡선에는 적용된다!

5.2.7 타원형 곡선

```
\setlength{\unitlength}{0.75cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}%
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}%
    {\line(1,0){6}}
  \thicklines
  \put(2,3){\oval(3,1.8)}
  \thinlines
  \put(3,2){\oval(3,1.8)}
  \thicklines
  \put(2,1){\oval(3,1.8)[tl]}
  \put(4,1){\oval(3,1.8)[b]}
  \put(4,3){\oval(3,1.8)[r]}
  \put(3,1.5){\oval(1.8,0.4)}
\end{picture}
```



다음 두 명령 중 하나로 중심 (x, y), 너비 w, 높이 h를 가지는 타원형 곡선(oval)을 그린다.

`\put(`x, y`){\oval(`w, h`)}`

또는

`\put(`x, y`){\oval(`w, h`)[`*position*`]}`

위치를 나타내는 옵션 인자 `t`, `b`, `l`, `r`은 각각 “top”, “bottom”, “left”, “right”를 나타내며, 예에서 보는 바와 같이 조합하여 쓸 수도 있다.

선의 두께는 두 종류의 명령으로 조정할 수 있다: `\linethickness{`*length*`}`를 사용하는 것이 그 하나이며 다른 방법으로는 `\thinlines`와 `\thicklines`를 사용한다. `\linethickness{`*length*`}`는 수직선과 수평선, 그리고 이차 베지어 곡선에만 적용되는 한편, `\thinlines`와 `\thicklines`는 경사 선분, 원과 타원에 적용된다.

5.2.8 그림 박스의 지정과 반복 사용

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(120,168)
\newsavebox{\foldera}
\savebox{\foldera}
  (40,32)[bl]{% definition
  \multiput(0,0)(0,28){2}
    {\line(1,0){40}}
  \multiput(0,0)(40,0){2}
    {\line(0,1){28}}
  \put(1,28){\oval(2,2)[tl]}
  \put(1,29){\line(1,0){5}}
  \put(9,29){\oval(6,6)[tl]}
  \put(9,32){\line(1,0){8}}
  \put(17,29){\oval(6,6)[tr]}
  \put(20,29){\line(1,0){19}}
  \put(39,28){\oval(2,2)[tr]}
}
\newsavebox{\folderb}
\savebox{\folderb}
  (40,32)[l]{%          definition
  \put(0,14){\line(1,0){8}}
  \put(8,0){\usebox{\foldera}}
}
\put(34,26){\line(0,1){102}}
\put(14,128){\usebox{\foldera}}
\multiput(34,86)(0,-37){3}
  {\usebox{\folderb}}
\end{picture}
```

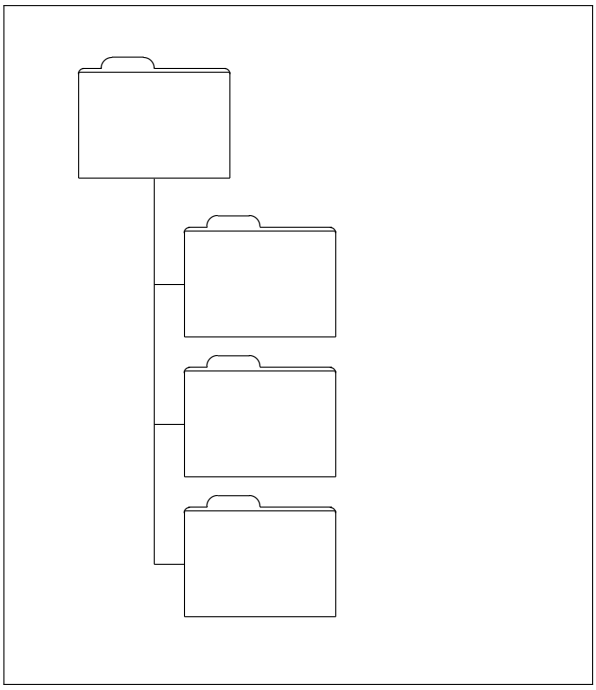


그림 박스(picture box)는 다음 명령으로 선언(*declare*) 하며

`\newsavebox{`*name*`}`

다음 명령으로 정의(*define*)하고

`\savebox{`*name*`}(`*width*`,`*height*`)[`*position*`]{`*content*`}`

마지막으로 다음의 명령으로 임의의 위치에 그리는 것이 흔히 쓰는 방법이다.

`\put(`x`,`y`){\usebox{`*name*`}}`

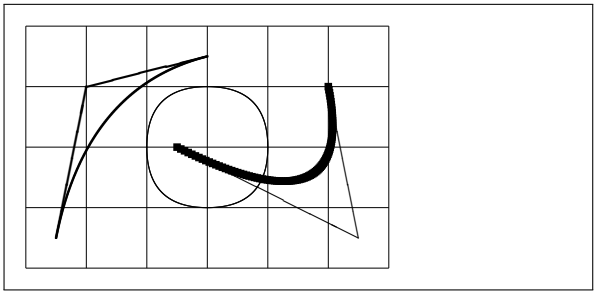
옵션 파라미터 *position*은 저장된 박스가 위치할 '고정점(anchor point)'을 정의한다. 위의 예에서는 `bl`로 설정되었고 이는 박스의 왼쪽 하단을 의미한다. 다른 위치 지정자로는 `top`과 `right`가 있다.

인자 *name*은 LaTeX의 저장소(storage bin)를 참조하며 따라서 이는 명령어로서의 성질을 가진다(이 때문에 위의 예에서와 같이 백슬래시(\)가 필요하다). 박스 내부 그림 안에 또 다른 박스의 그림을 넣을(nested) 수도 있다: 위의 예에서는 `\folderb`의 정의 내부에 `\foldera`가 사용되었다.

`\line` 명령은 선분의 길이가 대략 3 mm 이하면 작동하지 않기에 `\oval` 명령을 사용했다.

5.2.9 이차 베지어 곡선

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)
  \linethickness{0.075mm}
  \multiput(0,0)(1,0){7}
    {\line(0,1){4}}
  \multiput(0,0)(0,1){5}
    {\line(1,0){6}}
  \thicklines
  \put(0.5,0.5){\line(1,5){0.5}}
  \put(1,3){\line(4,1){2}}
  \qbezier(0.5,0.5)(1,3)(3,3.5)
  \thinlines
  \put(2.5,2){\line(2,-1){3}}
  \put(5.5,0.5){\line(-1,5){0.5}}
  \linethickness{1mm}
  \qbezier(2.5,2)(5.5,0.5)(5,3)
  \thinlines
  \qbezier(4,2)(4,3)(3,3)
  \qbezier(3,3)(2,3)(2,2)
  \qbezier(2,2)(2,1)(3,1)
  \qbezier(3,1)(4,1)(4,2)
\end{picture}
```



위의 예에서 볼 수 있듯이 하나의 원을 네 개의 이차 베지어 곡선(quadratic Bézier curve)으로 분할하는 것은 만족스럽지 않다. 적어도 여덟 개가 필요하다. 위의 그림은 `\linethickness`가 수직선과 수평선에 주는 효과와 `\thinlines`와 `\thicklines`가 경사선분에 주는 효과를 보여주기도 한다. 또한 위의 그림은 두 가지의 명령 모두 이차 베지어 곡선에 영향을 준다는 것을 보이는데, 나중에 적용된 명령이 이전의 명령을 덮어쓴다.

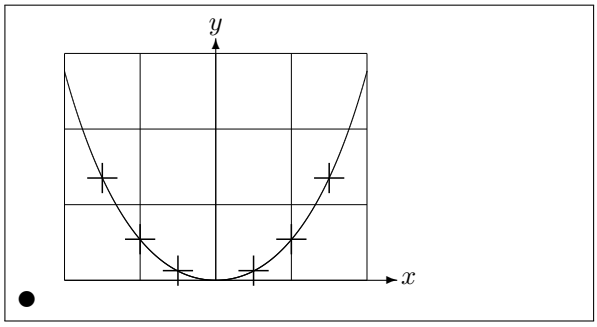
한 이차 베지어 곡선에서 $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$가 끝점을, m_1, m_2가 각각의 기울기를 나타낸다고 하자. 중간조정점(intermediate control point) $S = (x, y)$는 다음의 식에 의해 주어진다.

$$\begin{cases} x = & \dfrac{m_2 x_2 - m_1 x_1 - (y_2 - y_1)}{m_2 - m_1}, \\ y = & y_i + m_i(x - x_i) \qquad (i = 1, 2). \end{cases} \tag{5.1}$$

Graphics in LaTeX 2ε [16]에서 필요한 `\qbezier` 명령문을 도출하는 Java 프로그램을 볼 수 있다.

5.2.10 현수선

```
\setlength{\unitlength}{1cm}
\begin{picture}(4.3,3.6)(-2.5,-0.25)
\put(-2,0){\vector(1,0){4.4}}
\put(2.45,-.05){$x$}
\put(0,0){\vector(0,1){3.2}}
\put(0,3.35){\makebox(0,0){$y$}}
\qbezier(0.0,0.0)(1.2384,0.0)
  (2.0,2.7622)
\qbezier(0.0,0.0)(-1.2384,0.0)
  (-2.0,2.7622)
\linethickness{.075mm}
\multiput(-2,0)(1,0){5}
  {\line(0,1){3}}
\multiput(-2,0)(0,1){4}
  {\line(1,0){4}}
\linethickness{.2mm}
\put( .3,.12763){\line(1,0){.4}}
\put(.5,-.07237){\line(0,1){.4}}
\put(-.7,.12763){\line(1,0){.4}}
\put(-.5,-.07237){\line(0,1){.4}}
\put(.8,.54308){\line(1,0){.4}}
\put(1,.34308){\line(0,1){.4}}
\put(-1.2,.54308){\line(1,0){.4}}
\put(-1,.34308){\line(0,1){.4}}
\put(1.3,1.35241){\line(1,0){.4}}
\put(1.5,1.15241){\line(0,1){.4}}
\put(-1.7,1.35241){\line(1,0){.4}}
\put(-1.5,1.15241){\line(0,1){.4}}
\put(-2.5,-0.25){\circle*{0.2}}
\end{picture}
```



위의 예에서 현수선 $y = \cosh x - 1$의 대칭적인 반쪽을 각각 이차 베지어 곡선으로 근사시켰다. 오른쪽 반은 $(2,\ 2.7622)$에서 끝나고 기울기는 $m = 3.6269$이다. 식 (5.1)을 다시 이용하면 중간조정점을 계산해낼 수 있다. 이는 $(1.2384,\ 0)$과 $(-1.2384,\ 0)$이다. 십자 표시는 실제 현수선의 점들을 나타낸다. 오차는 1퍼센트보다 작아 거의 알아채기 힘들 정도이다.

이 예는 `\begin{picture}`의 옵션 인자가 사용되는 방식을 보여준다.

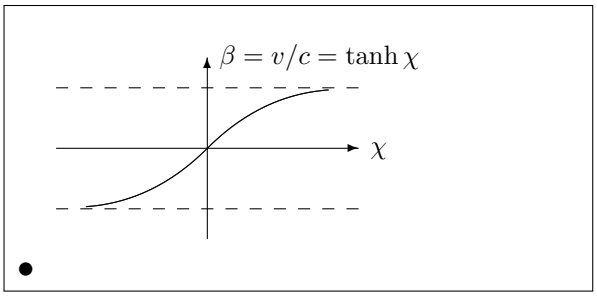
그림(picture)은 편리한 "수학적" 좌표로 정의되는데, 다음과 같이 표현하면

```
\begin{picture}(4.3,3.6)(-2.5,-0.25)
```

그림의 왼쪽 아래에 $(-2.5, -0.25)$의 좌표를 지정한다는 것을 의미한다(이는 위 그림에서 검은 원으로 표시되어 나타나 있다).

5.2.11 특수상대성이론의 속도

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(6,4)(-3,-2)
  \put(-2.5,0){\vector(1,0){5}}
  \put(2.7,-0.1){$\chi$}
  \put(0,-1.5){\vector(0,1){3}}
  \multiput(-2.5,1)(0.4,0){13}
    {\line(1,0){0.2}}
  \multiput(-2.5,-1)(0.4,0){13}
    {\line(1,0){0.2}}
  \put(0.2,1.4)
    {$\beta=v/c=\tanh\chi$}
  \qbezier(0,0)(0.8853,0.8853)
    (2,0.9640)
  \qbezier(0,0)(-0.8853,-0.8853)
    (-2,-0.9640)
  \put(-3,-2){\circle*{0.2}}
\end{picture}
```



식 (5.1)을 사용하여 두 베지어 곡선의 조정점들을 계산했다. 양의 방향 가지(positive branch)는 $P_1 = (0, 0)$, $m_1 = 1$ 그리고 $P_2 = (2, \tanh 2)$, $m_2 = 1/\cosh^2 2$에 의해 결정된다. 또 다시 그림은 수학적으로 편리한 좌표로 정의되었는데, 좌표 $(-3, -2)$가 왼쪽 아래의 끝점(검은 점)에 배정되었다.

5.3 PGF와 TikZ 그래픽 패키지

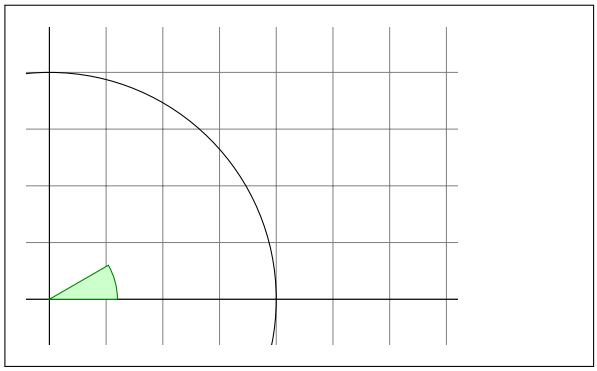
오늘날 모든 LaTeX 결과물 생성 시스템은 질 좋은 벡터 그래픽을 그려낼 수 있으며, 다만 그 인터페이스들이 다양할 뿐이다. pgf 패키지는 이 인터페이스들 위에 추상적인 레이어를 제공한다. pgf 패키지는 자체적으로 방대한 매뉴얼과 튜토리얼을 담고 있다. 따라서 이 작디 작은 절에서 우리는 그 패키지가 할 수 있는 일의 일부만을 볼 수 있을 뿐이다.

pgf 패키지는 tikz 패키지를 통해 하이 레벨 랭기지를 함께 제공한다. pgf는 도형을 그리는 엔진이고 TikZ는 그 엔진에 대한 사용자 인터페이스를 제공하는 고수준 언어라고 생각할 수 있다. TikZ는 사용자의 문서 내부에서 바로 그래프를 그려낼 수 있는 매우 효율적인 명령들을 제공한다. 사용자는 TikZ 명령들을 `tikzpicture` 환경 내에서 사용하면 된다.

위에서 언급했듯 pgf와 그 "친구들"을 다루기 위한 훌륭한 매뉴얼이 존재한다. 따라서 이들의 원리를 직접 설명하기보다는, 사용자에게 첫 인상을 심어주기 위해 이들이 어떤 결과물을 내놓는지를 몇 가지 살펴보기로 한다.

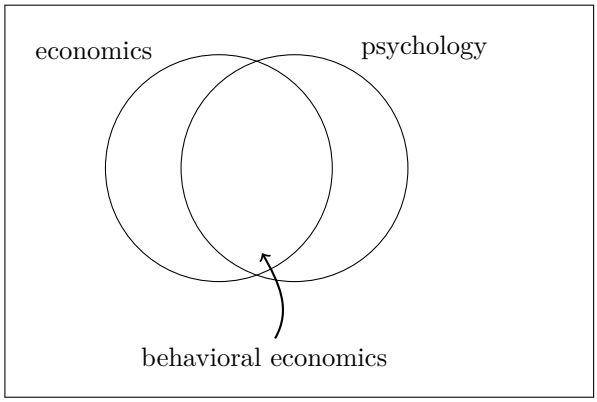
우선 간단한 nonsense diagram부터 보자.

```
\begin{tikzpicture}[scale=3]
  \clip (-0.1,-0.2)
     rectangle (1.8,1.2);
  \draw[step=.25cm,gray,very thin]
       (-1.4,-1.4) grid (3.4,3.4);
  \draw (-1.5,0) -- (2.5,0);
  \draw (0,-1.5) -- (0,1.5);
  \draw (0,0) circle (1cm);
  \filldraw[fill=green!20!white,
            draw=green!50!black]
    (0,0) -- (3mm,0mm)
         arc (0:30:3mm) -- cycle;
\end{tikzpicture}
```



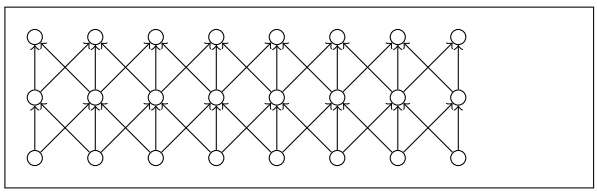
세미콜론(;)을 주목하라. 이것이 개별 명령들을 구분짓는다.
다음은 간단한 벤 다이어그램을 그리는 예이다.

```
\begin{tikzpicture}
  \node[circle,draw,
        minimum size=3cm,
        label=120:{economics}]
         at (0,0) {};
  \node[circle,draw,
        minimum size=3cm,
        label=60:{psychology}]
         at (1,0) {};
  \node (i) at (0.5,-1) {};
  \node at (0.6,-2.5)
    {behavioral economics}
    edge[->,thick,
         out=60,in=-60] (i);
\end{tikzpicture}
```



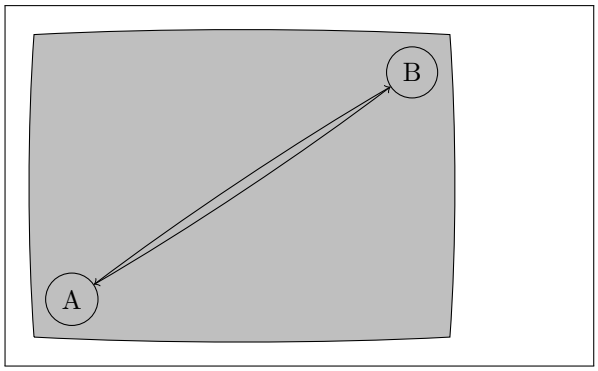
다음 예에서는 foreach 반복문을 주목하라.

```
\begin{tikzpicture}[scale=0.8]
  \tikzstyle{v}=[circle,
     minimum size=2mm,inner sep=0pt,draw]
  \foreach \i in {1,...,8}
    \foreach \j in {1,...,3}
      \node[v](G-\i-\j) at (\i,\j) {};
  \foreach \i in {1,...,8}
    \foreach \j/\o in {1/2,2/3}
      \draw[->]
        (G-\i-\j) -- (G-\i-\o);
  \foreach \i/\n in
    {1/2,2/3,3/4,4/5,5/6,6/7,7/8}
    \foreach \j/\o in {1/2,2/3} {
       \draw[->] (G-\i-\j) -- (G-\n-\o);
       \draw[->] (G-\n-\j) -- (G-\i-\o);
    }
\end{tikzpicture}
```

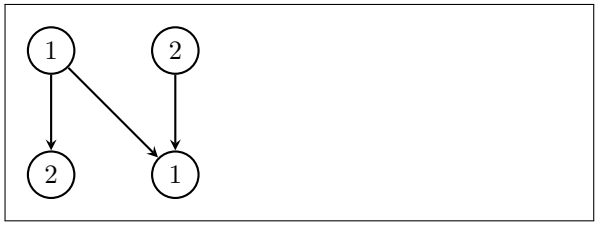


문서의 전처리부(preamble)에서 \usetikzlibrary로 새로운 기능을 제공하는 라이브러리(library)를 불러오면, 다음의 살짝 휜 박스와 같은 특별한 도형들을 그릴 때 필요한 다양한 기능을 추가할 수 있다.

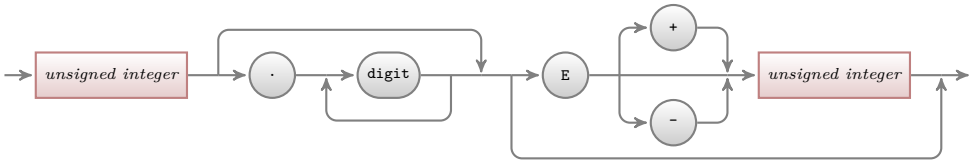
```
\usetikzlibrary{%
  decorations.pathmorphing}
\begin{tikzpicture}[
     decoration={bent,aspect=.3}]
 \draw [decorate,fill=lightgray]
        (0,0) rectangle (5.5,4);
 \node[circle,draw]
        (A) at (.5,.5) {A};
 \node[circle,draw]
        (B) at (5,3.5) {B};
 \draw[->,decorate] (A) -- (B);
 \draw[->,decorate] (B) -- (A);
\end{tikzpicture}
```




```
\usetikzlibrary{positioning}
\begin{tikzpicture}[xscale=6,
     yscale=8,>=stealth]
  \tikzstyle{v}=[circle,
     minimum size=1mm,draw,thick]
  \node[v] (a) {$1$};
  \node[v] (b) [right=of a] {$2$};
  \node[v] (c) [below=of a] {$2$};
  \node[v] (d) [below=of b] {$1$};
  \draw[thick,->]
        (a) to node {} (c);
  \draw[thick,->]
        (a) to node {} (d);
  \draw[thick,->]
        (b) to node {} (d);
\end{tikzpicture}
```



사용자는 파스칼 프로그래밍 책에서 바로 뛰쳐나온 것 같은 syntax 다이어그램조차 그릴 수 있다. 다만 그 코딩이 위의 예시에 비해 다소 무시무시하기 때문에 그 결과물만을 보기로 하자. pgf 문서를 찾아보면 이것과 동일한 다이어그램을 그리는 과정에 대한 세세한 설명을 찾을 수 있다.

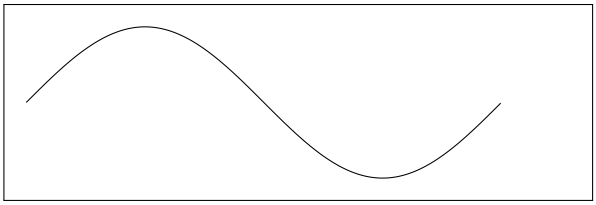


이게 끝이 아니다. 만약 사용자가 데이터를 플롯하거나 함수의 그래프를 그려야 한다면 pgfplot 패키지를 파고들어보면 좋다. 이 패키지는 플롯을 위해 필요한 모든 것을 제공하고 있으며, 더 나아가 외부 명령 gnuplot을 사용하면 함수를 직접 사용해 그래프를 그릴 수도 있다.

사실 pgfplot 패키지를 로드하지 않아도, 웬만한 함수를 플롯할 수 있다. 그 한 예를 들자면 다음과 같다.[4]

[4] [역주] 이 한 문단은 역자의 보충이다.

```
\begin{tikzpicture}
\draw plot [domain=0:6.28,samples=500]
  (\x,{sin(\x r)});
\end{tikzpicture}
```



더 많은 영감을 얻고자 한다면 Kjell Magne Fauske의 http://www.texample.net/tikz/에 방문해보기를 권한다. 이 사이트는 수많은 아름다운 그래프와 다른 LaTeX 코드들을 담고 있는 창고와 같으면서 이 순간에도 커지고 있다. TeXample.net의 list of tools to work with PGF/TikZ에서는 PGF/TikZ를 다룰 때 쓸 수 있는 도구들이 다수 수록되어 있기 때문에 모든 것을 직접 만들어내지 않아도 사용할 수 있다.

제 6 장

마음대로 바꾸기

지금까지 배운 명령을 이용하면 누구라도 납득할 만한 문서를 작성할 수 있다. 그다지 눈에 확 띄는 것은 아닐지 몰라도 좋은 조판 규칙을 잘 준수하고 있어서 읽기 쉽고 보기에도 좋은 문서가 될 것이다.

그러나 LaTeX이 제공하는 명령과 환경만으로는 만족하지 못하거나 꼭 필요한 일을 하는 데 이미 있는 명령만으로는 충분치 않은 상황이 있게 마련이다.

이 장에서는 LaTeX에게 새로운 일을 시키거나 기본적으로 주어지는 것과 다른 모양의 결과를 만들어내도록 하는 방법에 대해 약간의 힌트를 제공하려 한다.

6.1 명령, 환경, 패키지를 새로 정의하기

이 책자에서 명령을 소개할 때 명령어에 박스를 치고 책 뒤 색인에 나타나게 해놓았다. 이 일을 하기 위해 LaTeX 명령을 일일이 써넣지 않고 대신 이 목적에 맞는 명령과 환경을 정의했다. 그리고 이것을 모아 패키지를 만들었다. 이로써 간단히 다음과 같이 쓸 수 있다.

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

\dum

이 예제에 lscommand라는 새로운 환경과 \ci라는 새로운 명령을 쓰였다. lscommand는 명령어 주변에 박스를 그리는 일을 하며 \ci라는 명령은 명령어 이름을 식자하고 그것을 색인에 넣는 일을 한다. 이 책의 뒤에 있는 색인에서 \dum 명령이 나타나 있는지 찾아보라. \dum 항목이 나타나 있고 \dum에 대해 언급한 모든 페이지의 번호가 나와 있을 것이다.

만약에 명령어 이름에 박스를 치는 것이 싫증났다고 하자. 그러면 간단히 lscommand의 정의를 고치면 원하는 새로운 모양으로 바뀐다. 문서를 처음부터 뒤지면서 LaTeX 명

령어를 소개하려고 명령어 이름에 박스를 치고 있는 부분을 모두 찾아 하나하나 수정하는 것에 비하면 너무 쉽다.

6.1.1 새로운 명령

나만의 새로운 명령을 만들어보자.

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

기본적으로 이 명령은 두 개의 인자를 요구한다. 만들려고 하는 명령의 이름인 *name*과 그 **정의부**(*definition*)이다. 대괄호 안의 *num* 인자는 새 명령이 취할 (최대 9까지) 인자의 개수를 지정하는 것인데 선택 인자이므로 생략될 수 있다. 선택 인자를 주지 않으면 기본값인 0이 쓰인다. 즉 아무런 인자도 받아들이지 않는 명령이 된다.

다음 두 예제를 보면 이해하기 쉬울 것이다. 첫 번째 예제에서 새 명령의 이름이 `\tnss`라고 주어졌다. 이 책의 제목(The Not So Short Introduction to LaTeX 2_ε)의 약자인데 실제로 이 제목을 여러 번 반복해서 써야 할 일이 있으면 이렇게 정의한 명령이 꽤 쓸모있을 것이다.

```
\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”

그 다음 예제는 두 개의 인자를 받아들이는 새로운 명령을 정의하는 방법을 보여주고 있다. `#1` 표시한 것은 첫 번째 인자로, `#2`는 두 번째 인자로 치환된다.

```
\newcommand{\txsit}[2]
 {This is the \emph{#1}
  #2 Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
\item \txsit{not so}{short}
\item \txsit{very}{long}
\end{itemize}
```

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

이미 있는 명령을 `\newcommand`로 만드는 것은 허용되지 않는다. 이미 있는 명령을 수정하는 경우에 써야 하는 명령이 따로 있다. `\renewcommand`가 그것이다. 명령의 사용법은 `\newcommand`와 같다.

`\providecommand`라는 명령도 있다. 이것은 `\newcommand`와 비슷하지만 만약 같은 이름의 명령이 이미 정의되어 있다면 새로 정의하는 것을 무시한다. 즉 이미 정의된 같은 명령이 없다면 `\newcommand`하라는 의미이다.

LaTeX 명령에 뒤따르는 공백에 관해서 주의를 기울여야 하는 부분이 있다. 5페이지를 보라. 여기서 명령어 직후의 스페이스가 사라진다는 것을 배웠다. 인자를 주기 위한 중괄호가 있으면 그 뒤의 스페이스는 사라지지 않으므로 필요하다면 빈 인자`{}`로 스페이스를 살리는 방법에 대해서도 이미 배운 바 있다.

6.1.2 새로운 환경

새로운 명령을 만드는 \newcommand 명령이 있듯이 새로운 환경을 만드는 명령도 있다. \newenvironment가 그것이다. 사용법은 다음과 같다.

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

\newenvironment 명령도 인자들을 갖는다. *before* 인자로 전달되는 것은 환경 안의 텍스트보다 먼저 실행된다. *after* 인자로 오는 것은 \end{*name*}을 만났을 때 실행할 내용이다.

```
\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 선택인자는 \newcommand 때와 같은 방법으로 사용한다.

LaTeX은 이미 존재하는 환경과 같은 이름을 가진 환경을 새롭게 정의하지 못하게 하고 있다. 굳이 같은 이름의 환경을 만들어서 덮어써야 한다면 \renewenvironment 명령을 사용한다. \newenvironment와 문법은 동일하다.

이 예제에서 사용한 몇 가지 명령은 나중에 설명된다. 예를 들어 \rule 명령에 대해서는 109페이지에서 다룬다. \stretch에 대해서는 104페이지에서, 그리고 \hspace에 대해서는 104페이지에서 더 자세한 설명을 볼 수 있다.

6.1.3 불필요한 스페이스 없애기

새로운 환경을 정의하다보면 여분의 스페이스가 끼어들어서 고생하는 경우가 많다. 이런 것이 심각한 결과를 초래하기도 하는 것이다. 예를 들어보자. 다음과 같은 환경을 만들려고 한다. 의도는 현재 문단의 첫 들여쓰기를 없애고 환경이 끝난 다음 첫 문단의 들여쓰기도 없애고 싶다는 것이겠다.

```
\newenvironment{simple}%
 {\noindent}%
 {\par\noindent}

\begin{simple}
See the space\\to the left.
\end{simple}
Same\\here.
```

See the space
to the left.
Same
here.

무엇이 잘못되었는지 알겠는가? \begin 블록을 실행하는 과정에서 스페이스가 끼어들었고 \end 블록에서도 마찬가지인 것으로 보인다. \ignorespaces라는 명령을 두면 begin 블록을 실행하면서 끼어드는 스페이스를 무시하게 할 수 있다. 환경이 끝난 뒤의 스페이스를 무시하게 하는 것은 이것으로 바로 되지 않는다. 왜냐하면 LaTeX 환경이 종료되는

특별한 과정이 실행되기 이전까지의 스페이스만을 무시할 것이기 때문이다. 그래서 약간 트릭을 써야 한다. \ignorespacesafterend라는 명령은 특별한 환경의 끝내기 과정을 종료한 뒤에 \ignorespaces를 실행하라는 의미이다.

```
\newenvironment{correct}%
 {\noindent\ignorespaces}%
 {\par\noindent%
   \ignorespacesafterend}

\begin{correct}
No space\\to the left.
\end{correct}
Same\\here.
```

No space
to the left.
Same
here.

6.1.4 명령행 LaTeX

유닉스류의 운영체제를 사용중이라면 LaTeX 문서 작업을 위해 Makefile을 만들어서 쓰고 있을 것이다. 그런 조건에서라면 LaTeX에 명령행 파라미터를 주어서 같은 소스에서 다른 결과물을 얻어내게 할 수 있다.[1]

```
\usepackage{ifthen}
\ifthenelse{\equal{\blackandwhite}{true}}{
  % "black and white" mode; do something..
}{
  % "color" mode; do something different..
}
```

LaTeX을 다음과 같이 실행한다.

```
latex '\newcommand{\blackandwhite}{true}\input{test.tex}'
```

제일 먼저 \blackandwhite를 정의하고 그 다음에 입력 파일을 불러들였다. 만들어지는 결과물은 색상을 넣지 않은 흑백 버전이 될 것이다.

6.1.5 나만의 패키지

상당량의 새로운 명령과 환경을 정의하였다면 문서의 전처리부가 아주 길어졌을 것이다. 그렇다면 이 모든 새로 정의한 명령과 환경을 LaTeX 패키지로 만들면 좋지 않겠나 생각하게 된다. \usepackage 명령을 써서 패키지를 문서에 들여올 수 있다.

패키지를 작성하는 것은 기본적으로 문서 전처리부의 내용을 별도의 파일로 옮기는 것이다. 패키지 파일의 확장자는 .sty로 한다. 특별한 명령이 하나 있는데

\ProvidesPackage{*package name*}

패키지 시작 부분 맨처음에 써주는 것으로 패키지의 이름을 LaTeX에게 등록시키는 역할을 한다. 같은 패키지를 두 번 이상 포함하려 할 적에 이미 등록되었는지 여부를 LaTeX이

[1] [역주] Makefile과는 상관없이 윈도우즈 시스템에서도 cmd 명령행을 열고 아래 설명과 같은 일을 똑같이 할 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 6.1: 패키지 예제

체크하게 함으로써 같은 명령이 두 번 이상 정의되면서 발생하는 많은 에러를 피하게 해준다. 그림 6.1은 위에서 만들어 본 명령과 환경으로 작성한 간단한 패키지의 예제이다.

6.2 폰트와 크기

6.2.1 폰트 바꾸기 명령

LaTeX은 문서의 논리적 구조(장절 표제, 각주 등)에 따라 적절한 폰트와 그 크기를 선택한다. 그런데 폰트와 크기를 직접 바꾸고 싶은 경우가 있을 것이다. 표 6.1과 6.2에 열거된 명령을 사용할 수 있다. 각 폰트의 실제 크기는 디자인 문제로서 사용한 클래스와 옵션에 따라 달라진다. 표 6.3은 표준 클래스에 구현되어 있는 각 명령의 절대 크기를 포인트로 표시하였다.

```
{\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*.

한 가지 중요한 사실을 지적하자면 LaTeX에서 폰트 속성은 서로 독립적이라는 것이다. 크기나 폰트 종류를 변경하는 명령을 내리더라도 bold나 slant라는 속성은 여전히 유지된다.[2]

math mode 내에서 폰트를 바꾸려면 일시적으로 *math mode*를 빠져나가서 일반 텍스트로 입력해야 한다. 수식에 적용되는 폰트를 바꾸는 것은 다른 문제이며 별도의 명령을 사용한다. 표 6.4를 보라.

폰트 크기 명령 관련해서 중괄호(curly braces)가 아주 중요한 역할을 한다. 중괄호는 **범위**(*group*)를 설정하는 데 쓰인다. 대부분의 LaTeX 명령은 설정된 범위로 내에서 효력을 갖는다.

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

He likes large and small letters.

[2] [역주] 조금 기술적으로 말하면 폰트의 속성에는 family, series, shape, size가 있다. rm, sf, tt는 family이고 bf는 series이며 it, sl은 shape이다. 이들이 서로 독립적이라는 의미이다. 즉 family를 바꾸어도 series나 shape는 바뀌지 않는다.

표 6.1: 폰트

`\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 6.2: 폰트 크기

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

표 6.3: 표준 클래스의 폰트 크기 포인트

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 6.4: 수학 폰트

`\mathrm{...}`	Roman Font
`\mathbf{...}`	**Boldface Font**
`\mathsf{...}`	Sans Serif Font
`\mathtt{...}`	`Typewriter Font`
`\mathit{...}`	*Italic Font*
`\mathcal{...}`	$\mathcal{CALLIGRAPHIC\ FONT}$
`\mathnormal{...}`	*Normal Font*

폰트 크기 명령은 행간격도 변경한다. 그러나 명령이 영향을 미치는 유효 범위 안에서 문단이 종료될 때에만 그러하다. 그러므로 닫는 중괄호 }가 너무 일찍 놓이면 안 된다. 다음 두 예제에서 \par 명령의 위치에 따라 행간격이 달라지는 것을 주의깊게 보라.[3]

```
{\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.

전체 문단이나 여러 문단에 걸쳐 폰트 크기 명령을 적용하려 한다면 환경 형식으로 입력하는 것을 생각해볼 수 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 ...

여닫는 중괄호의 짝을 맞추기 위해 개수를 세는 노력을 줄일 수 있다.

6.2.2 폰트 명령 사용에 대한 중요한 경고

이 장을 시작하면서 지적한 바와 같이 이런 식의 명령을 사용하는 것은 문서를 망가뜨릴 수 있는 위험한 일이다. 왜냐하면 그것이 LaTeX의 기본 개념에 위배되기 때문이다. LaTeX은 문서의 논리적 마크업과 시각적 모양을 분리한다. 말하자면, 만약 특정 정보를 나타내기 위하여 폰트 모양을 바꾸어서 문서의 이곳저곳에 나타내어야 할 일이 있다고 하자. 이 때는 반드시 \newcommand로 그 "논리적 의미를 나타내는 명령"을 정의해서 써야지 폰트 바꾸는 명령을 직접 써서는 안 된다.

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

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

위의 예에서 \oops가 경고를 나타내기 위해 쓰였다고 하자. 만약 나중에 경고 표시를 위해 \textbf를 쓰는 대신 다른 모양으로 표현하고 싶어졌을 때, 이것을 폰트 모양 명령으로 표현해두었다면 문서 전체를 뒤져서 \textbf를 찾은 다음에 그것이 위험에 대한 경고로 쓰인 경우인지 아닌지를 일일이 따져서 수정해야 한다. 이것은 LaTeX의 방식이 아니다. 간단히 전처리부의 \oops 정의를 수정하여 한꺼번에 다른 모양으로 바꾸는 것이 훨씬 쉽고 LaTeX의 철학에 부합한다.

그러므로 뭔가를 강조하려 할 때 LaTeX에게 전달해주어야 할 것은 이 단어를 "강조한다"는 사실이지 그 단어의 "폰트를 이러저러하게 바꾸라"는 것이 아니다. \emph 명령은 문맥에 따라 사용가능하지만 폰트 변경 명령은 그렇지 않다.

[3] \par는 빈 줄 하나와 같다.

```
\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.

6.2.3 조언

폰트와 폰트 크기에 대해 알아보는 것을 마치기 전에 여기서 한 마디 조언을 남기려 한다.[4]

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

6.3 간격

6.3.1 행 간격

문서 전체에 대하여 행 사이의 간격을[5] 늘리고 싶다면 전처리부에 다음과 같이 늘리고자 하는 값(배수)을 지정한다.

```
\linespread{factor}
```

"한 줄 반" 행 간격에 `\linespread{1.3}`을, "두 줄" 행 간격에 `\linespread{1.6}`을 쓰도록 하라. 행 간격의 기본값은 "1"이다.[6]

[한국어판을 위하여 역자가 추가]
한글 문서는 라틴 문자 문서에 비해 행간을 넉넉하게 잡는다. 이 번역본의 경우 oblivoir 클래스의 기본값을 적용하였으며 그 값은 1.3 정도에 해당한다. kotex 패키지에 [hangul] 옵션을 주면 (다른 변경사항과 함께) 행간이 늘어나서 한글 문서에 적합하게 설정되는 것을 볼 수 있다.

`\linespread` 명령의 효과는 꽤 강력해서 유연성이 부족하다. 임의로 특정 부분의 행간격을 굳이 바꾸어야 할 때 다음 명령을 쓰는 것도 고려해볼 수 있다.

```
\setlength{\baselineskip}{1.5\baselineskip}
```

[4] [역주] 이 문장은 "더 많은 폰트를 사용할수록 문서가 더 읽기 쉽고 아름다워진다"고 되어 있지만 그 식자된 모양을 보면 역설적 표현임을 알 수 있다. 이 문장은 다음과 같은 의미로 이해하는 것이 옳다. "너무 많은 폰트를 이유없이 남용하면 문서는 읽기도 어려워지고 지저분해진다."

[5] [역주] 한 행의 베이스라인과 그 다음 행의 베이스라인까지의 거리를 "행송"이라 하고 행의 아래쪽 끝단과 다음 행의 위쪽 끝 사이의 간격을 "행간"이라고 한다. LaTeX의 line spacing은 행송에 해당하는 값이지만 이 용어를 엄격히 적용하지 아니하고 "행 간격"이라 표현하였다.

[6] [역주] LaTeX에서는 행 간격 배수가 1일 때 실제로는 행간이 살짝 주어진다. 이 때문에 배행간의 배수가 2가 되지 않는 것이다.

```
{\setlength{\baselineskip}%
           {1.5\baselineskip}
This paragraph is typeset with
the baseline skip set to 1.5 of
what it was before. Note the par
command at the end of the
paragraph.\par}
This paragraph has a clear
purpose, it shows that after the
curly brace has been closed,
everything is back to normal.
```

This paragraph is typeset with the baseline skip set to 1.5 of what it was before. Note the par command at the end of the paragraph.
This paragraph has a clear purpose, it shows that after the curly brace has been closed, everything is back to normal.

6.3.2 문단 모양

LaTeX에는 문단 레이아웃에 영향을 끼치는 두 가지 파라미터가 있다.

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

전처리부에 위의 코드를 두면 문단의 레이아웃을 변경할 수 있다. 위의 코드가 하는 일은 문단의 들여쓰기를 0으로 만들면서 문단 사이의 간격을 벌리는 것이다.

plus와 minus 부분은 주어진 값의 범위 안에서 필요하다면 문단 사이 간격을 늘리거나 줄일 수 있음을 TeX에게 알려주는 역할을 한다. 문단을 페이지에 적절하게 맞추기 위한 것이다.

유럽쪽 문헌에서 문단 사이에 간격을 두고 들여쓰기 하지 않는 식의 조판을 흔히 볼 수 있다. 그런데 이 명령은 목차에도 영향을 준다는 것을 기억하자. 목차 사이의 간격이 정상보다 느슨하게 식자된다. 이것을 피하려면 위의 명령을 전처리부에 두지 말고 문서 본문의 \tableofcontents 명령 뒤에 두는 방법이 있다. 그러나 아예 이런 것을 사용하지 않는 것도 좋은데 대부분의 전문서적은 문단 사이의 별도 간격 없이 들여쓰기하는 방법으로 조판되기 때문이다.

들여쓰기되지 않은 문단을 들여쓰게 하는

`\indent`

명령이 있다.[7] 이 명령은 당연히 \parindent가 0으로 설정되어 있지 않을 때만 효력이 있다.

특정 문단의 들여쓰기를 없애고 싶다면

`\noindent`

이 명령을 문단 첫머리에 둔다. 장절명령 없이 시작하는 첫 문단에 적용할 때 편리하다.

[7] 장절 표제를 식자한 직후 첫 문단은 들여쓰기되지 않는 것이 기본이다. 첫 문단에도 들여쓰기를 적용하려면 indentfirst 패키지를 로드하여야 한다.

6.3.3 수평 간격

단어와 문장 사이의 간격은 LaTeX이 자동으로 결정한다. 수평 간격을 추가하려면

`\hspace{`*length*`}`

줄의 끝이나 시작부분에도 간격이 유효하여야 한다면 `\hspace` 대신 `\hspace*`를 사용하라. *length*는 길이를 나타내는 숫자에 단위를 붙인 꼴이다. 길이 단위는 표 6.5에 정리해 두었다.

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

This is a space of 1.5 cm.

다음 명령은 특별한 가변폭 간격(rubber space)을 만들어낸다.

`\stretch{`*n*`}`

이와 같이 명령하면 한 행의 남은 공간이 모두 찰 때까지 간격을 채워넣는다. 같은 행 안에 `\hspace{\stretch{`*n*`}}` 명령이 두 번 이상 있으면 채울 수 있는 공간을 주어진 확장배수에 비례하여 차지한다.

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

x x x

수평간격과 텍스트를 함께 쓸 때 간격의 크기를 현재 폰트 크기에 맞추어서 지정할 수 있으면 좋을 것이다. 텍스트 폰트의 크기를 나타내는 상대적인 길이 단위 em과 ex를 사용하면 된다.[8]

```
{\Large{}big\hspace{1em}y}\\
{\tiny{}tin\hspace{1em}y}
```

big y
tin y

6.3.4 수직 간격

문단 사이의 간격, 장절 표제의 간격 등은 LaTeX이 스스로 결정한다. 만약 문단 사이에 추가 간격을 두어야 할 필요가 있다면 다음 명령을 쓴다.

`\vspace{`*length*`}`

이 명령 앞뒤에 보통 빈 줄을 두어야 한다. 페이지의 마지막이나 첫머리에도 간격이 유지되도록 하려면 `\vspace` 대신 `\vspace*`를 사용하라.

`\stretch` 명령을 `\pagebreak` 명령과 함께 써서 페이지의 하단이나 중앙에 한 줄이 오게 할 수 있다.

[8] [역주] em은 현재 폰트 M자의 폭이라는 뜻에서, ex는 x자의 높이라는 뜻에서 온 단위이다. 그렇지만 em은 실제로는 디자인 단위로서 10pt 폰트에서 1em은 10pt이다. 1ex는 Latin Modern Roman 폰트에서 약 4.3pt 정도이다.

표 6.5: TEX 단위

`mm`	millimetre ≈ 1/25 inch
`cm`	centimetre = 10 mm
`in`	inch = 25.4 mm
`pt`	point ≈ 1/72 inch ≈ $\frac{1}{3}$ mm
`em`	approx width of an 'M' in the current font
`ex`	approx height of an 'x' in the current font

```
Some text \ldots

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

같은 문단 내에서, 또는 표 안에서 두 줄 사이에 간격을 주려면 다음과 같이 한다.

`\\[`*length*`]`

`\bigskip`과 `\medskip`, `\smallskip`은 수직 간격 명령으로 미리 정의된 길이만큼 떨어뜨리는데 정확히 몇 포인트일지는 신경쓸 필요 없다.

6.4 페이지 레이아웃

LATEX에서는 용지 크기를 `\documentclass` 명령의 옵션으로 지정한다. 그러면 텍스트의 여백(margins) 크기를 자동으로 결정한다. 그러나 이렇게 만들어진 페이지가 마음에 들지 않을 수 있다. 당연히 바꾸는 것이 가능하다. 그림 6.2는 바꿀 수 있는 파라미터를 모두 보여준다. 이 그림은 'tools' 묶음의 layout 패키지로 그렸다.

잠깐! …… "문서의 판면 폭이 너무 좁으니 좀 넓히자"는 생각으로 뭔가를 하기 전에 조금만 더 생각해보라. 다른 것도 마찬가지지만 LATEX이 기본 페이지 레이아웃을 그렇게 설정하는 데는 이유가 있다.

확실히 (한때 쓰던) MS Word의 경우 판면이 이렇게 좁지 않았다. 그러나 실제 책[9]을 꺼내서 한 줄에 몇 글자나 들어가 있는지 세어보라. 각 행마다 대략 66자를 넘지 않을 것이다. 이제 기본적으로 만들어지는 LATEX 문서에 대해 역시 행당 글자수를 세어보면 마찬가지로 한 행에 66자 정도가 들어간다는 것을 알게 될 것이다. 경험상 한 줄이 너무 길어서 들어가는 글자수가 많으면 독서가 곤란해진다. 줄의 끝에서 다음 줄 처음으로 이동하는 거리가 길어서 눈을 이동시키기 더 어려워지기 때문이다.

그러므로 문서 본문 너비를 키우는 것은 독자에게 독서 부담을 가중하는 것이 된다는 점을 유념하여야 한다. 그럼에도 불구하고 꼭 해야 한다면 어떻게 하면 되는지 알려주겠다.

이 파라미터를 바꾸기 위한 명령을 두 가지 제공한다. 대개 전처리부에 두면 된다.

[9] 믿을 만한 출판사에서 출간된 진짜 책을 말한다.

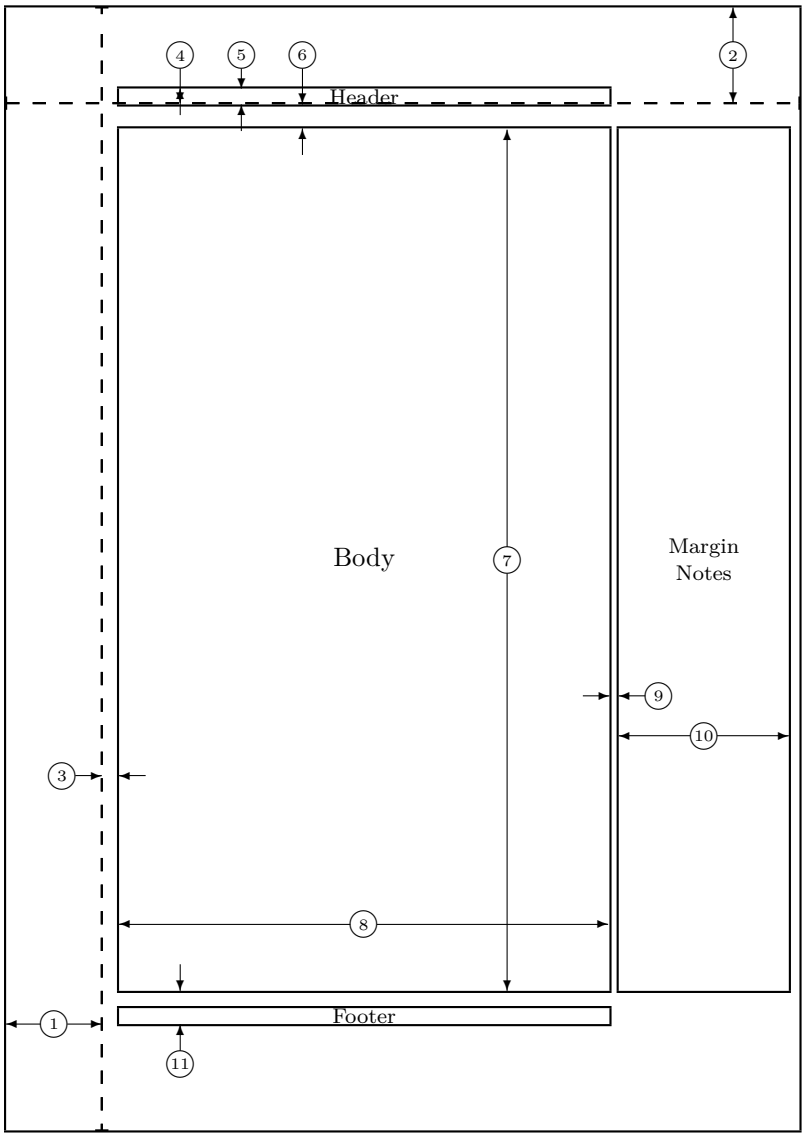


그림 6.2: 이 책의 레이아웃 파라미터. layouts 패키지를 이용하면 현재 문서의 레이아웃을 확인할 수 있다.

첫 번째 명령은 파라미터에 일정한 값을 부여하는 것이다.

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

두 번째 것은 파라미터에 일정 길이를 더하는 것이다.

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

두 번째 명령이 `\setlength`를 쓰는 것보다 유용하다. 이미 있는 값을 기준으로 상대적으로 설정할 수 있기 때문이다. 전체 문단 폭(text width)에 1cm를 더하려면 전처리부에 다음과 같이 선언하면 된다.

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

이런 종류의 일을 할 때에 calc 패키지가 도움이 된다. `\setlength` 명령의 인자 안에서 산술 연산을 할 수 있게 한다. 숫자가 필요한 다른 곳에서도 연산식 인자로 값을 입력할 수 있다.

6.5 길이 관련 재미있는 응용

가능한 한 LaTeX 문서에서는 절대 길이를 쓰지 않는 것이 좋다. 텍스트의 너비나 높이와 같은 다른 페이지 요소의 길이를 기준으로 상대 길이를 지정하는 것이 낫다. 예컨대 한 페이지 전체를 그래픽으로 채우려 할 때 그 폭(width)을 `\textwidth`로 설정한다.

다음 세 개의 명령은 텍스트 문자열(*text*)의 폭(width), 높이(height), 깊이(depth)를 계산하게 하여 변수(*variable*)에 할당하는 것이다.

`\settoheight{`*variable*`}{`*text*`}`
`\settodepth{`*variable*`}{`*text*`}`
`\settowidth{`*variable*`}{`*text*`}`

다음 예제에서 이 명령을 응용해보았다.

```
\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 adjacent 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 adjacent 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?

6.6 박스(Box)

LaTeX은 박스를 쌓아서 페이지를 만든다. 먼저 글자 하나하나가 작은 박스이다. 이 박스를 풀로 붙이듯이 이어붙여서 단어를 만든다. 그 단어를 다시 다른 단어와 이어붙이는데 이 때는 길이가 늘거나 줄어들 수 있는 특별한 연결요소(glue)를 풀처럼 사용한다. 그렇게 함으로써 한 줄이 판면에 꼭 맞도록 조절할 수 있다.

물론 이 설명이 실제 TeX이 하는 일에 비하면 너무 간략한 설명이라는 것을 안다. 그러나 핵심은 TeX이 박스와 글루를 가지고 작업한다는 점이다. 글자만이 박스가 아니다. 무엇이든지 박스에 들어갈 수 있다. 박스 안에 다른 박스가 들어갈 수도 있다. 일단 박스에 들어가고 나면 전체가 하나의 글자인 것처럼 LaTeX이 취급한다.

명시적으로 언급하지는 않았지만 이전 장에서 이미 박스를 다루어본 적이 있다. `tabular` 환경이나 `\includegraphics`가 그러한데 둘 다 하나의 박스를 만든다. 그러므로 표와 그림을 나란히 놓는 것도 매우 쉽다. 박스를 이어맞춘 길이가 본문 폭보다 크지 않게 하면 된다.

원한다면 하나의 문단도 박스 하나에 넣을 수 있다. 이를테면

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

이와 같이 `\parbox` 명령을 쓰거나

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

이런 식으로 `minipage` 환경을 쓰면 된다. `pos` 파라미터는 `c`, `t`, `b` 중에서 하나를 취하는데, 주변 텍스트 베이스라인을 기준으로 한 박스의 수직 정렬 위치를 나타낸다. `width`는 박스의 폭을 지시하는 길이이다. `minipage`와 `\parbox`의 차이점은 `\parbox` 안에는 사용할 수 없는 명령이나 환경이 있다는 것이다. 반면 `minipage` 안에는 무엇이든 가능하다.

`\parbox`가 줄나눔같은 수직적 조판요소까지 포함하여 문단 전체를 박스에 담는 데 비해 오직 수평적으로 나열된 것들만을 박스에 넣는 명령도 있다. 이미 그 중 하나를 써본 적이 있는데 `\mbox`가 그것이다. 이것은 이어지는 일련의 박스를 넣으면서 두 단어 사이에서 줄나눔이 일어나지 못하게 한다. 박스가 박스 안에 들어갈 수 있으므로 이 수평 박스를 유연하게 잘 이용할 수 있다.

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

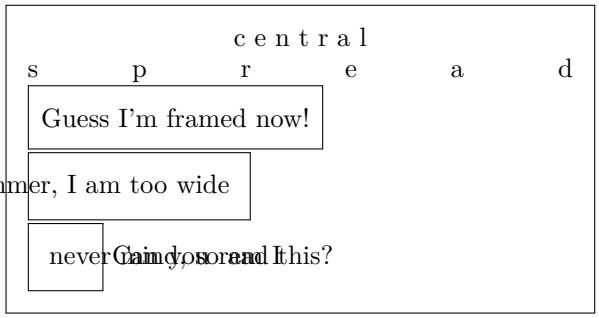
`width`는 박스의 폭을 나타내는데 박스 외부에서 보는 길이가 된다.[10] 이 파라미터에는 길이 표현이 올 수 있는 것은 당연하고 그밖에 `\width`, `\height`, `\depth`, `\totalheight`와 같은 명령이 올 수도 있다. 이들은 *text*로 주어지는 것을 조판하였을 때 가지게 되는 값을 측정하여 취한 것이다. *pos* 인자는 `c`, `l`, `r`, `s` 가운데 하나를 취할 수 있는데 각각 **c**enter, flush**l**eft, flush**r**ight, **s**pread의 의미이다. spread란 박스를 꽉 채우도록 텍스트를 벌려 배열하라는 뜻이다.

[10] 이 말은 박스 내부에 있는 것들의 길이보다 박스 길이가 작을 수도 있다는 뜻이다. 박스의 width를 0pt로 정의할 수도 있는데 그렇게 하면 주변 박스의 배열에 영향을 주지 않으면서 어떤 것을 식자할 수 있다.

\framebox라는 명령은 \makebox와 완전히 똑같이 동작한다. 그러면서 박스 주변에 선을 그려준다.

다음 예제를 살펴보면 \makebox와 \framebox 명령을 어떻게 쓰는지 짐작할 수 있을 것이다.[11]

```
\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?
```

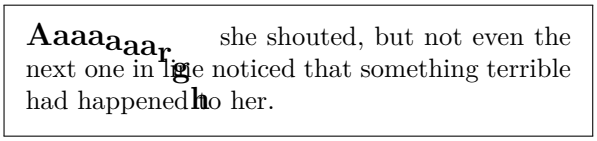


수평 제어에 대해 알아보았다. 다음 차례는 수직 제어에 대한 것이 당연하다.[12] 어려울 것이 없다.

\raisebox{*lift*}[*extend-above-baseline*][*extend-below-baseline*]{*text*}

이 명령은 박스의 수직 위치 이동을 정의한다. 앞에서부터 세 번째까지 인자에 \width, \height, \depth, \totalheight를 쓸 수 있다. 이들은 *text*로 주어진 문자열의 크기에 따라 결정되는 값이므로 박스 크기에 맞추어서 이동하게 할 수 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}}}
she shouted, but not even the next
one in line noticed that something
terrible had happened to her.
```



6.7 괘선(Rule)

몇 페이지 앞에서 본 적이 있는 명령이다.

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

보편적인 이용방법은 간단한 검은색 박스를 그리는 것이다. 또 다른 용법 중에 36페이지에서 이미 이 명령으로 "폭이 0인 괘선"을 strut로 이용하는 방법을 배운 적이 있다.

[11] [역주] 앞서 글자 박스가 단어가 될 때는 "풀"로 이어붙였고 단어와 단어 사이는 "늘어나는 길이(글루)"로 연결하였다고 한 말을 곰곰 생각해본다면 spread할 때 벌어질 수 있는 것은 단어와 단어 사이이지 글자와 글자 사이가 아님을 짐작할 수 있다. 다음에 보이는 예제에서 spread하기 위해서 글자 사이에 스페이스를 준 것은 그런 까닭이다.

[12] 완전 제어는 수평제어와 수직제어를 모두 달성해야 성취된다.

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

수평괘선이나 수직괘선을 그릴 때 좋다. 이 책자의 표지에 그려진 괘선이 이 명령으로 작성되었다.

끝

부록 A

LaTeX 설치하기

Knuth가 TeX 소스를 공개했을 때는 오픈소스나 자유 소프트웨어에 대해 아무도 알지 못하던 시절이었다. TeX에 부과된 라이센스는—Knuth 자신이 제공하는 일련의 테스트를 통과하지 못한다면 그것을 TeX이라고 부를 수 없다는 점을 제외하면—그 소스를 가지고 원하는 모든 것을 다 할 수 있게 한 것이었다. 그것은 지구상의 거의 모든 플랫폼에서 TeX이 구현되어 자유롭게 활용할 수 있게 되는 결과를 가져왔다. 이 부록에서는 리눅스, 맥 오에스, 윈도우즈에서 TeX을 설치하고 설정하는 문제에 관해 다루고자 한다.

A 설치해야 할 것

LaTeX을 운영하고자 한다면 설치해야 할 프로그램이 여럿 있다.

1. 소스 파일을 처리하여 PDF나 DVI로 조판하는 TeX/LaTeX 프로그램
2. LaTeX 소스 파일을 편집하기 위한 텍스트 에디터. 어떤 제품은 에디터 내에서 LaTeX 프로그램을 불러서 구동할 수 있다.
3. PDF/DVI 화면보기 프로그램. 문서를 미리보기하고 인쇄할 수 있게 한다.
4. 문서에 포함할 PostScript와 이미지 파일을 다루는 프로그램

모든 플랫폼에 이 요구를 충족하는 프로그램들이 다양하게 존재한다. 여기서는 우리가 잘 알고 좋아하면서 많이 쓰고 있는 것에 대해서만 언급하겠다.

B 크로스 플랫폼 에디터

TeX 자체는 여러 다양한 플랫폼에서 운영되고 있지만 LaTeX 에디터는 상당히 오랜 기간 플랫폼 종속적이었다.

지난 수 년 간 저자는 Texmaker를 즐겨 써왔다. 내장 pdf 프리뷰와 구문 하일라이팅을 갖춘 유용한 에디터이기도 하지만 특히 윈도우즈, 맥, 유닉스/리눅스에서 모두 똑같이 잘 실행된다는 장점이 있다. 자세한 사항은 http://www.xm1math.net/texmaker를 보라. TeXstudio라는 Texmaker의 파생 버전도 있다 (http://texstudio.sourceforge.net/). 이 또한 잘 관리되고 있으며 세 가지 주요 운영체제에서 모두 이용가능하다.

아래의 OS관련 절에서 플랫폼에 특정한 에디터에 대해서 언급할 것이다.

C 맥 OS의 TeX

C.1 TeX 배포판

MacTeX을 다운받으면 된다. LaTeX 전체 설치 이외에 상당한 추가 도구를 제공한다. http://www.tug.org/mactex/에서 얻을 수 있다.

C.2 맥 OS TeX 에디터

여러 플랫폼에서 운영되는 Texmaker를 제안하는 바(부록 B절) 지만 만족하지 못하겠다면…….

가장 유명한 맥의 오픈소스 LaTeX 에디터는 TeXshop이다. http://www.uoregon.edu/~koch/texshop에서 얻을 수 있으며 MacTeX 배포판에도 포함되어 있다.

최근의 TeX Live는 TeXShop 디자인을 흉내낸 다중플랫폼 편집기 TeXworks 에디터 (http://texworks.org)를 포함한다. TeXworks는 Qt 툴킷을 사용하기 때문에 이를 지원하는 어떤 플랫폼(맥 OS, 윈도우즈, 리눅스)에서도 쓸 수 있다.

C.3 PDFView를 사용해보자

LaTeX이 생성한 PDF 파일을 보는 데 PDFView를 사용해보라. LaTeX 텍스트 에디터와 긴밀하게 통합된다. PDFView는 오픈소스 응용 프로그램으로서 http://pdfview.sourceforge.net/에서 얻을 수 있다. 설치 후에 환경설정으로 가서 *automatically reload documents*를 활성화하는 것과 PDFSync 지원이 적절하게 잘 되어 있는지 확인하는 것을 잊지 말자.

D 윈도우즈의 TeX

D.1 TeX 얻기

먼저 http://www.miktex.org에서 MiKTeX이라는 훌륭한 배포판을 다운로드받는다. 여기에 LaTeX 문서를 컴파일하는 데 필요한 기본 프로그램과 파일이 전부 들어 있다. 저자가 보기에 이 배포판의 가장 멋진 점은 문서를 컴파일하는 도중에 필요한 패키지를 즉시 다운로드하여 자동으로 설치해주는 것이다. 한편 TeX Live 배포판은 윈도우즈, 유닉스, 맥 OS용이 있으므로 이것을 선택해도 좋다. http://www.tug.org/texlive를 방문해보라.

D.2 LATEX 에디터

여러 플랫폼에서 운영되는 Texmaker를 제안하는 바(부록 B절)지만 만족하지 못하겠다면…….

TeXnicCenter는 프로그래밍의 세계에서 가져온 많은 개념을 활용하여 윈도우즈에서 최선의 효율적인 LATEX 저작 환경을 제공한다. http://www.texniccenter.org/에서 얻을 수 있다. TeXnicCenter는 MiKTEX과 잘 어울린다.

최근 TEX Live 배포판에 포함되어 있는 TEXworks 에디터가 있다. http://texworks.org. 유니코드를 지원하며 최소 Windows XP를 요구한다.

D.3 문서 보기 프로그램

DVI 문서 프리뷰로서 MiKTEX이 설치해주는 Yap을 사용한다. PDF 뷰어로는 SumatraPDF http://blog.kowalczyk.info/software/sumatrapdf/를 추천한다. 이 뷰어를 사용하면 소스 코드의 특정 위치에 대응하는 PDF상의 위치로 즉시 이동하는 것이 가능하다.

D.4 그림 관련

LATEX에서 고품질 그래픽으로 작업하려면 Encapsulated PostScript (eps)나 PDF를 그림 포맷으로 해야 한다. 이런 그래픽은 GhostScript라는 프로그램의 도움을 받아야 할 경우가 많다. 자체 프론트엔드인 GhostView와 함께 http://www.cs.wisc.edu/~ghost/에서 얻을 수 있다.

사진이나 스캔본 같은 비트맵 그래픽이라면 Photoshop 대안인 Gimp를 고려해볼 수 있다. http://gimp-win.sourceforge.net/에서 다운로드 가능하다.

E 리눅스의 TEX

리눅스에서는 LATEX 사용 환경이 이미 마련되어 있는 것이나 마찬가지다. 설치 설정 과정이 자연스럽게 이루어진다. 패키지 매니저로 설치할 수 있는 패키지가 다음과 같다.

- texlive – TEX/LATEX 설치
- emacs (AUCTeX) – LATEX에 특화되어 있는 AUCTeX 패키지를 얹은 에디터
- ghostscript – PostScript 프로그램
- xpdf – pdf 미리보기 프로그램
- imagemagick – 비트맵 그림의 변환 도구
- gimp – 자유 소프트웨어로서 Photoshop에 대응
- inkscape – 자유 소프트웨어로서 Illustrator 또는 corel draw에 대응

윈도우즈에 가까운 GUI 에디터 환경을 원한다면 Texmaker를 고려해보라. B절을 볼 것.

대부분의 리눅스 배포판은 TEX 환경을 여러 개의 많은 개별 패키지로 나누어서 원하는 부분만 설치할 수 있게 하고 있다. 그러므로 만약 뭔가 빠진 것이 있으면 이 부분을 점검해보기 바란다.

문헌 목록

[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] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion,* (*2nd Edition*). Addison-Wesley, Reading, Massachusetts, 2004, ISBN 0-201-36299-6.

[4] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The LaTeX Graphics Companion.* Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4.

[5] LaTeX을 설치한 시스템 관리자는 *LaTeX Local Guide*라는 문서를 작성하여 제공한다. 여기에는 로컬 시스템에 특정한 사항이 설명되어 있어야 한다. 이 파일은 `local.tex` 이라는 이름으로 되어 있을 것이다. 불행히도 시스템 관리자가 게을러서 이 문서를 제공하지 않을 수도 있다. 그럴 때는 주변의 LaTeX 도사에게 물어보는 수밖에 없다.

[6] LaTeX3 Project Team. *LaTeX 2ε for authors.* LaTeX 2ε 배포판에 포함되어 있음. `usrguide.tex`.

[7] LaTeX3 Project Team. *LaTeX 2ε for Class and Package writers.* LaTeX 2ε 배포판에 포함되어 있음. `clsguide.tex`.

[8] LaTeX3 Project Team. *LaTeX 2ε Font selection.* LaTeX 2ε 배포판에 포함되어 있음. `fntguide.tex`.

[9] D. P. Carlisle. *Packages in the 'graphics' bundle.* 'graphics' 묶음에 포함되어 있음. `grfguide.tex`.

[10] Rainer Schöpf, Bernd Raichle, and Chris Rowley. *A New Implementation of LaTeX's verbatim Environments.* 'tools' 묶음에 포함되어 있음. `verbatim.dtx`.

[11] Vladimir Volovich, Werner Lemberg, and LaTeX3 Project Team. *Cyrillic languages support in LaTeX.* LaTeX 2ε 배포판에 포함되어 있음. `cyrguide.tex`.

[12] Graham Williams. *The TeX Catalogue* is a very complete listing of many TeX and LaTeX related packages. CTAN://help/Catalogue/catalogue.html에서 구할 수 있음.

[13] Kristoffer H. Rose. *Xy-pic User's Guide.* Xy-pic CTAN 배포처에서 다운로드 가능함.

[14] John D. Hobby. *A User's Manual for METAPOST.* http://cm.bell-labs.com/who/hobby/에서 다운로드 가능함.

[15] Alan Hoenig. *TeX Unbound.* Oxford University Press, 1998, ISBN 0-19-509685-1; 0-19-509686-X (pbk.)

[16] Urs Oswald. *Graphics in LaTeX 2ε*, containing some Java source files for generating arbitrary circles and ellipses within the `picture` environment, and *METAPOST - A Tutorial*. http://www.ursoswald.ch에서 둘 다 다운로드 가능.

[17] Till Tantau. *TikZ & PGF Manual*. CTAN://graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf에서 다운로드 가능.

[18] François Charette. *Polyglossia: A Babel Replacement for XeLaTeX*. TeX Live에 `polyglossia.pdf`로 포함되어 배포됨. (명령행에서 `texdoc polyglossia` 명령을 내리면 읽을 수 있음.)

[19] François Charette. *An ArabTeX-like interface for typesetting languages in Arabic script with XeLaTeX*. TeX Live에 `arabxetex.pdf`로 포함되어 있음. (명령행에서 `texdoc arabxetex`이라고 명령하면 읽을 수 있음.)

[20] Will Robertson and Khaled Hosny. *The `fontspec` package*. TeX Live에 `fontspec.pdf`로 포함되어 있음. (명령행에서 `texdoc fontspec`이라고 명령하면 읽을 수 있음.)

[21] Apostolos Syropoulos. *The `xgreek` package*. TeX Live에 `xgreek.pdf`로 포함되어 있음. (명령행에서 `texdoc xgreek`이라고 명령하면 읽을 수 있음.)

[22] Vafa Khalighi. *The `bidi` package*. TeX Live에 `bidi.pdf`로 포함되어 있음. (명령행에서 `texdoc bidi`라고 명령하면 읽을 수 있음.)

[23] Vafa Khalighi. *The `XePersian` package*. TeX Live에 `xepersian.pdf`로 포함되어 있음. (명령행에서 `texdoc xepersian`이라고 명령하면 읽을 수 있음.)

[24] Wenchang Sun. *The `xeCJK` package*. TeX Live에 `xeCJK.pdf`로 포함되어 있음. (`texdoc xecjk`를 명령하면 읽을 수 있음.)

[25] Dohyun Kim, 『XeTeX-ko 간단 매뉴얼』, TeX Live에 `kotex` 패키지와 함께 배포됨. (명령행에서 `texdoc xetexko`를 명령하면 읽을 수 있음.)

색인

역자후기

거의 15년만에 lshort을 다시 번역하여 내게 되었다. 이 책자는 수십 년에 이르는 유구한 시간의 시련을 견뎌내며 LaTeX 입문서로서 그 성가를 쌓아왔다. 일독을 권한다. 입문자라면 예제를 중심으로 조금 시간을 내어 공부한다면 얻는 것이 적지 않을 것이다.

지금은 각국 언어로 번역이 이루어져 CTAN에서 찾아볼 수 있지만 한국어판은 다른 언어와 비교해도 상당히 이른 시기에 번역본이 나왔고 CTAN에 올라가 있은 지도 오래되었다. 그리고 그 기간 갱신이 이루어지지 않아서 유감스럽게 생각하다가 이번에 새 판을 내게 되어 매우 흡족하다.

두 번에 걸친 lshort 한국어판 번역 작업은 한글 LaTeX의 발전 과정과 그 궤를 같이해 왔다. 2002년에는 HLaTeX으로, 2005년에는 dhucs로 작업하였으니 실로 격세지감이 있다 하겠다. 이제 koTeX과 XeLaTeX이 널리 쓰이게 되어 이런 훌륭한 결과물을 비교적 고생하지 않고 간단히 얻을 수 있게 된 것은 특기하여야 할 일일 것이다. 한글 LaTeX 개발을 위해 헌신한 모든 분들께 감사 인사를 드린다.

이번 번역은 이 책이 실용적인 한글 입문서가 되어야 한다는 관점에서 작업이 이루어졌다. 몇 가지 지난 번역본의 역어를 포기하지는 않았지만 지나친 순정주의를 지양하고 쉽게 읽히면서 LaTeX을 배우는 데 실제 도움이 될 만하게 만들려고 노력하였다.

역자들은 세 가지 방식으로 본문에 개입하였는데 (1) 방대한 역주를 달아서 독자의 이해를 도우려 하였으며, (2) 필요하다면 본문에 추가해 넣는 것도 마다하지 않았다. 그러나 본문의 추가는 최소한으로 억제하려 하였고 원문을 충분히 존중하면서 훼손하지 않으려고 애썼다. (3) 한국어 문서 작성과 관련된 두세 개의 소절을 추가하였다. 이 세 가지 역자의 개입은 보는 즉시 바로 알아볼 수 있도록 표지를 붙여두었다.

한국어판은 oblivoir 클래스를 바탕으로 작성하였으며 원본의 이곳저곳에 있는 명령, 환경, 설정의 많은 부분을 재구현하였다.

초고본을 KTUG 게시판에 공개하고 의견을 구하는 과정에 참여해주신 분들께 감사드린다. 좋은 문서가 되는 것은 독자의 기여가 없이 불가능한 것이라고 믿는다. 특별히 몇 번에 걸쳐 읽으면서 수많은 오자를 꼼꼼하게 잡아내고 좋은 표현을 제안해준 이주호, 이호재, 윤석천 제씨의 노고에 깊은 감사를 드린다.

우리가 들인 시간과 노력이 LaTeX에 입문하는 분들께 조금이라도 얻게 해주는 바가 있다면 큰 기쁨이겠다.

2019년 2월
역자를 대표하여 김강수

한글 폰트에 관하여

이 문서의 배포판에서 영문자는 원본과 동일하게 Latin Modern 폰트를 이용하였다. 한글은 KoPub World 폰트를 사용하였는데 이는 한국출판인회의에서 전자출판진흥사업의 일부로 무료 공개한 서체이다. http://www.kopus.org/biz/electronic/font.aspx에서 얻을 수 있다.

2005년판 4.17의 역자후기

KTUG이 생기기 전에 필자의 개인 홈페이지에서 처음 번역이 이루어진 lshort-kr은 그 후 CTAN에 등록되어 우리글로 된 LaTeX 입문서로서 많은 분들에게 읽혀왔다. 지금 돌이켜보면 당시 HLaTeX을 기반으로 번역 작업을 하면서 고생했던 것은 영문을 옮기는 것도 옮기는 것이지만 LaTeX에서 한글을 구현하는 것 자체가 쉽지 않았던 면이 있었다.

KTUG이 이루어놓은 놀라운 업적들이 이 번역본에 고스란히 담겨 있다. 우선 최종 출력물인 PDF의 품위 자체가 달라졌다. 한글 책갈피, 텍스트의 추출 · 검색, 하이퍼링크 등, 3.20판을 번역할 당시에는 잘 상상하기 어렵던 일들이 너무나 손쉽게 가능해졌다. 또한 유니코드 기반의 한글로 이행함으로써 더이상 HLaTeX의 EUC-KR 한계를 걱정하지 않아도 되는 행복한 상황에서 4.17판을 번역하게 되었다. 이 한글판 lshort-kr은 unicode/dhucs 패키지를 이용하여 한글을 구현하였다.

기술적 뒷받침이 이루어졌으므로 이제는 내용의 질을 제고할 때라고 생각한다. 이번 번역도 자원한 분들과 함께 공동작업으로 이루어졌다. 영어에 능한 분들과 언어학을 전공하신 분까지 번역에 합류함으로써 이제는 번역의 질에 있어서도 부끄럽지 않을 정도가 되었다고 생각한다.

이번에 공동번역자들이 합의한 번역 원칙은 다음과 같다.

- 영문을 그대로 옮겨놓는 번역을 피하고 실제 LaTeX에 입문하는 초보자들이 쉽게 이해하고 적용할 수 있도록 문장을 완전히 새로 쓴다. 필요하다면 내용을 보충하거나 생략할 수 있다. 중요한 것은 입문서로서 이 책의 전달력이지 원문의 충실한 재현에 있지 않다. 즉, 번역문에서 어떤 영문 문장의 기미도 발견할 수 없도록, 아름다운 우리말로 이루어진 문서를 만드는 것이 목표이다. (우리말식 용어를 사용해야 한다는 뜻은 결코 아님).
- 예제는 이번에도 영문을 그대로 노출시킨다. 한글 LaTeX의 발전에 힘입어 예제를 한글화하는 것이 어려운 일은 아니지만 여전히 이 책은 LaTeX 사용법에 관한 책이다. 그러나 한글 LaTeX 사용환경이 현저히 다르거나 한글화와 관련한 중요한 사항이 있을 때는 이에 대한 언급을 별도의 절이나 역주로 만들어 붙인다.
- 용어의 통일은 중요하다. 역어의 선택은 되도록이면 우리말화하되 우리말 용어 자체를 지나치게 중시한 나머지 도무지 알 수 없거나 머리 속에서 영어로 다시 옮겨야 이해가 되는 것을 배제하고 적당한 우리말이 없는 경우에는 차라리 흔히 쓰이는 영어식 표현을 그대로 쓴다.

모든 경우에 이 원칙이 완전히 관철되지는 않았을 것이다. 그러나 번역자들이 지향하는 번역의 방향은 대체로 일치하였다고 생각한다.

한국어판이 개선되지 않는데도 불구하고 메일링 리스트에서 삭제하지 않고 꾸준히 훌륭한 문서를 만들어 온 저자 Tobias Oetiker 씨에게 감사한다. 원래 한국어판 번역본에 실렸던 글을 더 확장하고 보충하여 영문판에 실리도록 글을 써주신 신정식 님께도 감사드린다. 이 번역본에 새로이 수정하여 실린 "한국어 지원" 절은 lshort의 다음 판에서 이용할 수 있도록 Oetiker 씨에게 수정본을 보낼 생각이다. 사실상 KTUG의 활동을 가능케 한 조진환 박사, KTUG에 깊은 애정을 보이시는 지도자이자 후원자이신 남상호 박사, dhucs와 dhhangul의 저자로서 한글화에 중요한 기여를 하신 김도현 교수, KTUG의 중심인 이주호 님, 이호재 님, KTUG Collection을 가능하게 한 홍석호 님, 그밖에 KTUG과 lshort-kr에 관심과 격려를 보내주신 모든 질문자와 답변자 분들께 감사드린다. 이주호 님은 번역본 전체를 읽으면서 주교정자로서의 역할을 담당해주셨다. 특별히 교정·교열 과정에서 긴 교정표를 작성해주신 이상직 교수, 그리고 번역본을 읽고 의견을 제시해주신 딸기아빠 님과 커꿈 님께도 감사드린다. 번역본이 더 읽고 이해하기 쉬워진 것은 전적으로 이 분들의 공이다.

번역이 이루어질 무렵 한글날을 맞아 한겨레신문사에서 한겨레결체를 공개한 것은 공개 한글 글꼴이 부족한 TEX 공동체에 좋은 선물이었다. 이 문서는 한겨레결체를 본문 글꼴로 채용하였다.

이 번역본에 잘못이 있다면 책임은 김강수에게 있다. 공동 역자들은 오역에 대하여 책임이 없다. 초보자분으로서 이 책에서 조금이라도 얻은 것이 있다면 역자들에게 알려주기 바란다. 개선을 위한 의견, 오류의 지적도 환영한다. KTUG 게시판이나 전자우편을 이용할 수 있다.

함께 고생한 공동 역자들의 노력에 감사하고 행운을 빌면서.

번역자 : 김강수, 이기황, MIKA, 샘처럼, 김지운.

공동번역자를 대표하여... 김강수

2002년판 3.20판의 역자후기

이 책은 사용법이 쉽지만은 않은 LATEX의 입문서로 이미 정평이 있다.

당연히 LATEX 2ε를 제대로 이해하고 쓰기 위해서는 *The LATEX Companion* [3]이 있어야 할 것이고, 좀더 고급의 독자들은 *The TEXbook*을 보아야 하겠지만, 논문 작성 등 일반적 용도에는 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 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, 3.1의 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.

각 장별로 최종적으로 사용된 텍스트의 초벌 역자와 교열자는 다음 표와 같다.

감사의 말, 서문	강윤배·장대훈	김강수
제1장	김강수	장대훈·김강수
제2장	강윤배	장대훈·김강수
제3장	장대훈	김강수
제4장	김재우*	김강수
제5장	장대훈	김강수
참고문헌	장대훈	김강수

* 장대훈 님의 번역본도 있었는데, 김재우 님의 텍스트를 주로 살리면서 장대훈 님의 번역을 참조하여 교열하였음.

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.

번역상 주의한 것은 다음과 같다.

- 예제들은 영문을 그냥 노출시켰다. 이렇게 한 이유는, 이 책이 LaTeX 2_ε에 대한 설명이지 HLaTeX에 대한 설명이 아니라는 점 때문이었다. 다시 말하면 이 예제들을 한글화했을 때, 그것은 LaTeX 2_ε를 통해 실행되는 것이 아니라 HLaTeX을 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 공동번역자들이 통일하기 위해서 여러번 시도했지만 완전히 일치하지는 못했을 수도 있을 것이다. 이 문제는 차차 고쳐가겠다.
- 최근 CTAN의 디렉토리 구조가 바뀌면서 이 책에 나오는 URL 정보가 달라진 것이 있어서 그것을 바로잡았다.

책을 옮기는 일은 솔직히 말하면 쉽지 않았다. 그 과정에서 격려해 준 김도현 님, 이현호 님, '무식인' 님을 비롯한 모든 분들에게 특별히 감사의 말을 전한다. 초벌 번역본과 교열본을 읽고 어색한 표현을 지적해 준 LaTeX을 전혀 모르는 나의 학생들에게도 고맙다는 인사를 전한다. HLaTeX의 저자인 은광희 님께 감사한다. 한글LaTeX이 없었으면 이 글의 번역은 불가능했을 것이다.

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

번역자 : 김강수, 강윤배, 장대훈, 김재우, 이재승, 현범석, 주철