The Memoir Class

The Memoir Class

for

Configurable Typesetting

User Guide

Peter Wilson

김강수 옮김

THP
The Herries Press
KTUG Documentation Team, trans.

The Herries Press, Normandy Park, WA.

Printed in the World

The paper used in this publication may meet the minimum requirements of the American National Standard for Information Sciences — Permanence of Paper for Printed Library Materials, ANSI Z39.48–1984.

10 09 08 07 06 05 04 03 02 01 15 14 13 12 11 10 9

First edition:	3 June 2001
Second impression, with corrections:	2 July 2001
Second edition:	14 July 2001
Second impression, with corrections:	3 August 2001
Third impression, with minor additions:	31 August 2001
Third edition:	17 November 2001
Fourth edition:	16 March 2002
Fifth edition:	10 August 2002
Sixth edition:	31 January 2004

memoir, *n.* a written record set down as material for a history or biography: a biographical sketch: a record of some study investigated by the writer: (in *pl.*) the transactions of a society. [Fr. *mémoire* — L. *memoria,* memory — *memor*, mindful.]

Chambers Twentieth Century Dictionary, New Edition, 1972.

memoir, *n.* [Fr. *mémoire,* masc., a memorandum, memoir, fem., memory < L. *memoria,* MEMORY] **1.** a biography or biographical notice, usually written by a relative or personal friend of the subject **2.** [*pl.*] an autobiography, usually a full or highly personal account **3.** [*pl.*] a report or record of important events based on the writer's personal observation, special knowledge, etc. **4.** a report or record of a scholarly investigation, scientific study, etc. **5.** [*pl.*] the record of the proceedings of a learned society

Webster's New World Dictionary, Second College Edition.

mem · oir [mémwɑːr, –wɔːr] *n.* ① **전기**, 실록; (고인의) 언행록; (보통 *pl.*) 추억의 기록, 회상[회고]록, 자서전. ② 연구 보고, 논문 (monograph); (*pl.*) 학회지, 기요(紀要). – · ist 회고록 집필자.

민중서림, 《엣센스 영한사전》, 탁상판 제*1*쇄.

차 례

그림 차례

표 차례

머리말

comp.text.tex 뉴스그룹을 지켜본 결과와 개인적인 경험을 통해 보면 LaTeX을 사용하는 데 있어서 문제는 주로 문서 디자인에 관한 것이다. ctt에 올라오는 질문들은 벌써 수년 전에 누군가가 답변을 작성하여 코드를 제시해 두었는데 이것을 반복해서 묻고 또 묻는 것들이 대부분이다. 최근에 작성된 답변은 '—— 패키지를 사용하라'는 형식으로 답변이 작성되는 것이 많은데 이 또한 반복해서 질문과 답변이 이어지고 있다.

나 역시 이 많은 패키지를 사용해왔다. 그러나 파일들이 항상 제 자리에 있는 것도 아니었으며 다양한 사용자 매뉴얼을 쉽게 찾을 수도 없었다. 나 자신이 작성한 패키지 역시 마찬가지였다. memoir 클래스는 LaTeX book 클래스에 디자인에 관련된 패키지들을 통합하려는 시도로 작성된 것이다. book 클래스를 선택한 이유는 report 클래스가 abstract 환경을 가지고 있다는 점만을 제외하면 book과 동일한 것으로 생각되었기 때문이다. abstract가 필요하다면 구현하는 것은 쉬운 일이다. 조금만 손을 보면 book 클래스 문서가 article 클래스 문서로 만든 것처럼 보이게 하는 것도 가능하다. memoir는 이런 여러 가지 점을 염두에 두고 만들어졌다.

memoir 클래스는 여러 외부 패키지를 이용하여 하던 일들을 하나의 클래스로 효과적으로 합친 것이다. 그렇지만 클래스 코드는 각 패키지가 제공하는 것과 동일한 기능을 수행하도록 새로이 작성되었다. 나 자신이 작성한 코드는 간단히 복사하여 붙인 경우도 있기는 하다. LaTeX과 그밖의 여러 훌륭한 패키지들이 없었다면 memoir는 불가능했을 것이다.

내가 사용한 패키지 이외에도 참고 문헌에 열거된 다른 패키지로부터 다양한 아이디어를 활용하였다. 그 저자들은 부지불식간이라 해도 어떤 식으로든 기여가 있었다. comp.text.tex 뉴스그룹 참여자들 또한 가치있는 역할을 하였는데 LaTeX으로 어떤 일을 하여야 하는지를 질문하는 것, 거기에 답변을 제공하는 것이 모두 자극이 되었다. 이 뉴스그룹은 대단히 친절하고 교육적인 포럼이다.

Peter Wilson
Seattle, WA
June 2001

초판 서문

이 책은 LaTeX 사용법 안내서가 아니라 memoir 클래스가 표준 LaTeX book 및 report 클래스와 어떤 점이 다른지를 설명하는 것이다. LaTeX자체를 소개하는 책은 매우 많으며 그 일부는 참고 문헌에 열거되어 있다. Lamport [Lam94]는 LaTeX의 오리지널 사용자 안내서이고 컴패니언 시리즈, 예컨대 [GMS94]는 이것을 더 확장하여 자세히 취급하고 있다. Comprehensive TeX Archive Network (CTAN) 은 공개 정보와 LaTeX 시스템 자체에 대한 중요한 원천이다. 일반적인 궁금증에 대해서는 FAQ [FAQ]를 보라. 이곳에서 특정 정보를 얻으려면 어디를 보아야 하는지를 가르쳐 줄 것이다. 특히 *The Not So Short Introduction to LaTeX2e* [Oet]과, Keith Reckdahl의 *Using imported graphics in LaTeX2e* [Rec97], Piet van Oostrum의 *Page layout in LaTeX* [Oos96] 등은 한번 읽어볼 만한 글들이다. 또 LaTeX에서 다양한 폰트를 사용하는 문제에 대해서는 내가 아는 한 Alan Hoenig의 책 [Hoe98]이 가장 훌륭한 안내서이다.

이 매뉴얼의 제1부는 책 디자인과 타이포그라피의 몇 가지 측면에 대한 간략한 논의로 시작한다. 여기서는 특정 타입세팅 도구에 대해서 언급하지 않는다. 이 주제에 관하여 참고 문헌에 열거된 책 중에서 내가 특히 좋아하는 것은 Bringhurst의 *The Elements of Typographic Style* [Bri92]이다. 제 I 부의 페이지 디자인은 LaTeX의 book 클래스 디자인을 이용하였고 이것이 memoir의 기본 디자인이다.

제2부는 memoir 클래스가 구현하고 있는 디자인에 대하여 좀더 자세한 내용을 다룬다.

memoir 클래스는 두 가지만을 제외하고 표준 LaTeX book 클래스의 기능을 모두 구현하고 있다. 그 두 가지는 다음과 같다.

- LaTeX v2.09의 옛 폰트 명령 — `\rm` (roman), `\tt` (`typewriter`), `\sf` (sans), `\bf` (**bold**), `\sl` (*slanted*), `\it` (*italic*), `\sc` (SMALL CAPS) — 이들은 지원되지 않는다. 사용되면 에러 메시지를 출력한다.

  `\em` (강조) 명령은 지원되지 않는 것은 아니나 경고 메시지를 출력한다.

- 타이틀 만들기 명령이 없다. 그 까닭은 타이틀 페이지가 책마다 별도로 디자인되어야 하는 것이기 때문이다. 필요하다면 titling 패키지 [Wil01g]의 타이틀 만들기 명령을

사용하면 된다. 이 패키지는 표준 LaTeX클래스의 타이틀 만들기 명령보다 확장된 기능을 제공한다.

이 점을 제외하면 이 클래스가 book 클래스가 제공하는 모든 기능을 다 제공하기를 희망한다.

이 클래스가 제공하는 추가 기능 중에서 중요한 것은 다음과 같다.

- 9, 10, 11, 12, 14 포인트 글꼴 크기 옵션
- 페이지, 텍스트, 여백 크기를 직관적으로 통제할 수 있는 도구
- 페이지 자름 표지
- 정말로 필요하다면 옛날 타자기 시절의 타입세팅(2배줄간격, 왼쪽맞춤, 하이픈 없음, 타자기 폰트)을 흉내낼 수 있음
- 절 헤딩 설정. 기정의 장 헤딩 양식과 장 헤딩을 정의하고 수정할 수 있는 편리한 방법.
- '무기명' 절 구분 (예: 빈 줄 또는 장식 괘선 구분)
- 페이지 면주 설정 기능. 다양한 기정의 페이지 스타일과 손쉬운 재정의.
- 캡션 설정 기능. 다양한 기정의 캡션 스타일과 손쉬운 재정의 도구.
- 새로운 리스트 정의 및 새로운 플로트를 'List of...'와 연동시키는 기능.
- 'List of...', 참고문헌목록, 색인 등을 설정하는 기능. 이들이 목차에 나타남.
- 에피그라피 지원

이밖에도 소소하지만 유용한 기능을 여러 가지 확인하게 될 것이다.

제 II 편에서 표준 LaTeX 페이지 스타일과 타이틀 붙이기 스타일을 만들기 위해 memoir를 어떻게 수정할 수 있는지도 시험한다.

글자체

이 안내서에서 사용될 글자체는 다음과 같다.

- LaTeX 클래스와 패키지의 명칭은 현재 글꼴로 표시한다.
- 클래스 옵션은 이 폰트로 표시한다.
- 장 스타일과 페이지 스타일의 명칭은 이 글자체로 표시한다.
- `LaTeX 코드는 이 글꼴로 표시한다.`

유의사항

현재 이 매뉴얼과 클래스 코드는 베타 테스트 상태이다. 즉 모든 기능이 의도한 대로 작동하기를 바라지만 에러가 발생할 가능성은 남아 있으며 설명은 빈약하고 빠진 항목도 있을 것이다. 새로운 릴리스가 있게 되면 현재 상태와 호환되지 않는 경우도 있을 수 있음을 알아두도록 하자.

특히 에러나 잘못된 설명 그리고 개선할 점에 대하여 건설적인 조언을 해주실 분들[1]에게 감사드린다. 저자에게 연락하려면 comp.text.tex에 포스팅하거나 peter.r.wilson@boeing.com에게 직접 전자우편을 보내면 된다.

TEX은 주로 많은 수학식이 포함된 문서를 조판하기 위해 개발되었다. 이런 저작물에서는 페이지 상의 텍스트 흐름이 수학식에 의하여 나누어지고 수학식이 들어갈 임의의 수직 간격이 확보되어야 한다. 그러나 대부분의 비전문 서적의 경우에는 고정된 간격으로 조판이 이루어지고 텍스트에 임의의 간격이 추가되지 않는 경우가 많다.

TEX은 이러한 임의의 간격 삽입을 아주 훌륭하게 해낸다. 즉 편집 영역의 높이가 고정되어 있더라도 약간의 간격을 줄이거나 늘려서 넣어주는 것이다. 그러므로 TEX에 기초하고 있는 LATEX역시 고정 간격 조판을 행하지 않는다. 이러한 고정 간격 조판을 해내는 것은 중요한 일인데 내가 아는 한 성공적인 것을 본 적이 없다.

이 안내서에는 많은 LATEX 명령의 이름이 포함되어 있고 이 명칭은 중간에서 끊을 수 없다. 그래서 그 가운데 일부는 여백 영역까지 삐져나가 있다. 이것을 제거하려면 한 행의 마지막에 그런 명령이 오지 않도록 문장을 다시 써야 한다. 이것은 지루한 일이고 안내서를 개정할 생각이므로 그냥 내버려두었다.

[1] 다음 분들에게서 값진 조언을 얻었다. Javier Bezos (jbezos@wanadoo.es), Sven Bovin (sven.bovin@chem.kuleuven.ac.be), Scott Pakin (pakin@uiuc.edu), Paul Stanley (pstanley@essexcourt-chambers.co.uk) .

재판 서문

초판을 낸 이래 memoir 클래스는 몇 가지 변화와 확장이 이루어졌다. 변화는 예기치못한 오류들을 제거하는 것이으로 매뉴얼의 내용에 일치하게 되었고 확장은 충분히 설명되지 못한 것이었다.

주요한 확장과 변화는 다음과 같다.

- subfigure 패키지를 함께 사용할 때 subfigure 옵션이 내부적으로 잘못 작동하는 것[2]을 수정하였다.
- article 옵션을 주면 article 클래스로 작성한 것을 흉내내도록 하였다.
- titling 패키지 [Wil01g]로부터 코드를 가져와 통합하였다(article 옵션을 지원한다).
- abstract 패키지 [Wil01a]로부터 코드를 가져와 통합하였다(article 옵션을 지원한다).

초판의 ToC, LoF, LoT표제를 수정하는 방법에 관한 설명은 완전히 잘못된 것이었고, `\newlistof` 매크로에 대한 설명도 잘못되어 있었다. 클래스의 코드를 수정하지는 않았지만 매뉴얼의 설명을 실제에 맞게 고쳤다. 원본을 쓰면서 몇 마디를 추가해야 했다.

그밖의 사소한 변경과 확장[3]이 이루어졌다. 초판과 비교해보면 알 수 있을 것이다.

클래스가 `verse` 환경을 제공하고 있었지만 초판에는 운문 조판에 관한 언급이 없었다. 시는 읽는 사람의 감정에 호소하기 위해서 특별한 모양으로 조판된다. verse 패키지 [Wil01k]가 시를 조판하는 데 도움이 될 것이다.

[2] Ignasi Furió Caldentey가 발견한 것이다.

[3] Kevin Lin과 Adriano Pascoletti에게 특히 감사한다.

제 3 판 서문

이 매뉴얼의 제2판이 간행된 이후 memoir 클래스가 베타 코드에서 정식 릴리스 판으로 업그레이드되었다. 클래스 코드와 매뉴얼 양자 모두에 확장이 이루어졌다.

주요한 변경과 확장은 다음과 같다.

- openleft 옵션으로 새 장을 짝수쪽에서 시작하는 것이 가능하게 되었다.
- verse 패키지 [Wil01k]에서 코드를 빌어와서 통합하였다. 운문 조판이 훨씬 유연해졌다.
- `\makepshook` 매크로를 `\makepsmarks`로 바꾸었다. 이전 버전과 호환성이 없다는 데 유의하라.
- 매뉴얼의 제1부를 재배열하고 확장하였다. 특히 조판 사례를 제공하기 위해 노력하였다.
- 코드와 매뉴얼의 사소한 불일치를 제거하였다. 코드가 매뉴얼에 적힌 대로 작동하도록 수정을 가하였다.

그밖에도 여러 가지 수정과 확장이 가해졌다.[4] 제2판과 비교해보면 알 수 있을 것이다.

[4]Ignasi Furió Caldentey, Daniel Richard G., Vladimir Ivanovic에게 특히 감사한다.

제 4 판 서문

매뉴얼의 제3판이 간행된 이후 memoir 클래스가 1.0에서 1.1 버전으로 업그레이드되었다. 클래스와 이 매뉴얼을 모두 수정하였다.

중요한 변화는 다음과 같다.

- subfigure 옵션은 불필요하여 제거하였다.
- subfloat가 클래스에 추가되었다. Steve Cochran은 자신의 subfigure 패키지 코드를 이 클래스에서 사용하는 것을 허락해주었다.
- 아직도 예전 LaTeX 2.09 버전의 폰트 명령을 여전히 사용하고 있는 패키지가 많다. 내키지 않지만 폐기된 예전 폰트 명령을 사용할 수 있도록 하는 옵션을 도입하였다.
- 이 클래스는 이제 natbib 패키지 [Dal99]와 잘 조화된다.
- 언제나처럼 코드와 매뉴얼의 불일치를 제거하였다. 매뉴얼에서 설명하는 것이 실제로 구현될 수 있도록 그 사이의 갭을 제거하고자 하였다.

더 많은 수정과 확장[5]이 이루어졌다. 이전 판과 비교해보면 알 수 있을 것이다.

[5]특히 William Adams, Ignasi Furió Caldentey, Steven Douglas Cochran, Henrik Holm, Rolf Niepraschk에게 감사한다.

제 5 판 서문

제4판이 간행된 이후 memoir 클래스가 1.1에서 1.2 버전으로 업그레이드되었다. 클래스와 이 매뉴얼 양자 모두에 변경이 가해졌다.

중요한 확장과 변경은 다음과 같다.

- 본문 활자 크기 옵션이 확장되었다.[6] 17pt 옵션이 추가되었다.
- 폰트 크기 명령(예, `\Large`)에 대응하는 폰트 크기 일부가 변경되어 사이즈 간의 변화폭을 균일하게 하였다.
- verbatim 텍스트를 포함하는 박스가 페이지가 넘어갈 때 나누어지게 하였다.
- verbatim 텍스트를 처리하는 여러 가지 방법을 제공한다. 각주도 verbatim 텍스트를 포함할 수 있다.
- 각주 조판을 제어할 수 있게 하였다. 불행히도 이를 위하여 thanks 노트 스타일을 처리하는 방법을 변경해야 하였다.
- comment 환경.
- 편리한 파일 입출력 방법을 제공한다.
- `\provide...` 명령을 추가하였다.
- `description` 환경을 수정하여 표준 환경과 비슷한 모양이 되게 하였다. 원래 memoir 클래스가 채택하였던 형태는 `blockdescription` 환경으로 이름이 변경되었다.
- `\chapter` 명령에 추가 옵션이 설정되었다. `\chapter*` 명령으로 헤더 텍스트를 설정할 수 있다.

늘 그렇듯이 코드와 매뉴얼의 사소한 불일치를 제거하였다. 새로운 문제가 들어오지 않았기를 바란다.

[6] 이것은 아이들이 학교 숙제를 하기 위해서 LaTeX을 사용한다는 Vittorio De Martino의 요청에 의한 것이다.

제 6 판 서문

제5판을 낸 이후로 memoir 클래스가 1.2버전에서 1.6버전까지 업그레이드되었다. 새로운 기능이 추가되었기 때문에 안내서 또한 이러한 추가를 반영하여 업데이트하였다.

주요한 확장과 변화는 다음과 같다.

- 배열과 표 조판 기능이 확장되었다. continuous tabular, automatic tabular 기능이 추가되었다.
- 각주 스타일과 다중 각주 기능이 확장되었다.
- 1단 색인 및 다중 색인 등 색인만들기가 확장되었다.
- 절단 표지(crop mark) 기능이 확장되었다.
- `\tableofcontents` 등 목차류를 여러 번 반복사용할 수 있도록 하였다.
- 섹션 타이틀에 대한 타이틀 참조가 가능해졌다.
- 용지번호 및 마지막 페이지와 마지막 용지 번호에 대한 접근이 가능하다.
- 숫자 형식 기능이 추가되었다.
- chapterbib, natbib 패키지와의 충돌이 개선되었다.

이밖에도 사소한 추가가 이루어졌다. 오류는 코드와 매뉴얼 양자 모두에서 되도록 제거하려고 애썼다. 새로운 오류가 들어오지 않았기를 바란다.

memoir 클래스와 그 사용설명서에 많은 분들이 기여해주셨다. 코드, 해법, 새로운 기능에 대한 제안은 물론이고 코드와 매뉴얼의 오류와 부적절한 점에 주의를 기울이도록 이끌어준 분들은 물론이고 가벼운 격려조차 많은 힘이 되었다. 나는 다음 분들에게 매우 감사한다. 그들이 알든 모르든 간에. Paul Abrahams, William Adams, Donald Arseneau, Jens Berger, Karl Berry, Javier Bezos, Sven Bovin, Alan Budden, Ignasi Furió Caldenty, Ezequiel Martín Cámara, David Carlisle, Steven Douglas Cochran, Michael Downes, Victor Eijkhout, Danie Els, Robin Fairbairns, Simon Fear, Kai von Fintel, Daniel Richard

G, Romano Giannetti, Kathryn Hargreaves, Sven Hartrumpf, Florence Henry, Cartsten Heinz, Peter Heslin, Morton Høgholm, Henrik Holm, Vladimir Ivanovich, Stefan Kahrs, Jøgen Larsen, Kevin Lin, Matthew Lovell, Daniel Luecking, Lars Madsen, Vittorio De Martino, Frank Mittelbach, Rolf Niepraschk, Patrik Nyman, Heiko Oberdiek, Scott Pakin, Adriano Pascoletti, Robert, Chris Rowley, Bernd Raichle, Doug Schenck, Rainer Schöpf, Paul Stanley, Reuben Thomas, Bastiaan Niels Veelo, Emanuele Vicentini, Jürgen Vollmer, 기타 여러분.

만약 실수로 어떤 분을 여기서 언급하지 않았다면 죄송할 따름이고 이 누락을 교정할 수 있도록 나에게 알려주기 바란다.

당연히, Donald Knuth의 TEX 시스템과 Leslie Lamport 이후 LATEX의 발전이 없었더라면 이 모든 일이 가능하지 않았을 것이다.

용어

다른 모든 전문 분야에서 그러하듯이 타이포그라피 다자인이나 인쇄업에 있어서도 자신들만이 사용하는 특별한 용어들이 있다.

먼저 면(page), 내지(leaf),[7] 용지(sheet)라는 명칭이 있다. 종이를 재단하여 묶어서 책을 만들었을 때 책 속의 종이 한 장을 가리켜서 내지(leaf)라 한다. 재단 전 종이를 용지*(stock)*라고 부르겠다. 한 장의 내지는 앞뒤 양면을 갖는다. 페이지란 내지의 한 면을 가리킨다. 책을 폈을 때 두 장의 내지를 만나게 된다. 오른쪽 내지의 전면을 우측면(recto)이라 한다. 왼쪽은 내지의 후면을 보게 되는데 이를 좌측면(verso)이라 한다. 따라서 한 장의 내지에도 우측면과 좌측면이 있으며 각각 내지의 전면과 후면이 된다. recto 페이지는 홀수 쪽번호가 붙고 verso 페이지는 짝수 쪽번호가 붙는다.

면주(folio)라는 용어도 있다.[8] 페이지 번호를 나타내는 타이포그라피 용어가 *folio*이다. '셰익스피어의 첫번째 중철본 책'이라고 할 때의 'folio'는 책의 크기를 가리킨다. 한편 '*folio* signature'라고 할 때는 인쇄된 용지를 접는 횟수를 가리키는 용어이다. 이 말과 혼동하여서는 안된다. 책의 모든 페이지에 면번호를 두지는 않는다. 면주가 전혀 없는 페이지도 있을 수 있다. 면번호 혹은 면주를 가진 페이지는 인쇄된 것이건 아니건 간에 페이지매김*(pagenation)*이 된다. 면 번호를 세지 않는 페이지에는 면주도 붙지 않는다.

폰트*(font)*는 문자의 집합이다. 금속활자나 납활자 시대에는 폰트라는 말이 특정 크기의 완전한 알파벳과 부가 문자 한 벌을 가리키는 말이었다. 오늘날에 와서는 크기와 관계없이 완전한 한 벌의 글자 세트를 가리키는 용어가 되었다. 로마자 폰트는 대문자(CAPITAL LETTERS), 작은 대문자(SMALL CAPITAL), 소문자(lowercase letters), 숫자, 문장부호, 리거쳐(합자. 'fi'나 'ffi'와 같은 것을 가리킨다) 및 약간의 특수 문자(& 등)로 이루어진다. 폰트 패밀리*(font family)*는 로만 폰트와 이탤릭 폰트 등과 같이 함께 어울리도록 디자인된 일군의 폰트 세트를 가리키는 말이다.

폰트 크기는 포인트 단위로 표시된다(72.27 포인트=1 인치=25.4밀리미터). 이 사이즈

[7] 내지는 보통 '장' 또는 '속장'이라고 한다—역주.

[8] 번역에서는 folio를 면주 또는 면번호라고 번역하였다. 면주는 柱(하시라), 면번호는 농브르(nombre)를 가리키는 말로 썼다. 주로 페이지 번호만을 찍어주는 것을 농브르라고 하고 장절 표제 등을 면의 header/footer에 표시하는 것과 농브르를 모두 합쳐서 하시라라고 한다. folio는 면번호를 말하거나 면주를 모두 가리키는 말로도 쓰인다—역주.

표 1: 인쇄 길이 단위

Name (abbreviation)	Value
포인트 point (pt)	
파이카 pica (pc)	1pc = 12pt
인치 inch (in)	1in = 72.27pt
센티미터 centimetre (cm)	2.54cm = 1in
밀리미터 millimetre (mm)	10mm = 1cm
큰포인트 big point (bp)	72bp = 72.27pt
디도 포인트 didot point (dd)	1157dd = 1238pt
시세로 cicero (cc)	1cc = 12dd

는 가장 긴 글자의 높이를 가리킨다고 할 수 있겠으나 폰트가 다르면 같은 사이즈라도 실제 높이가 다 다르다.

타이포그라피와 인쇄업에서 행과 행 사이의 수직 간격을 나타내는 말은 행간*(leading)*이다.[9] 이것도 포인트 단위로 표시되는데 폰트 크기보다 조금 크다. 통상 폰트를 표시하는 방법은 폰트 사이즈와 행간 크기를 슬래시로 구분하여 나타내는 것이다. 예를 들어 10/12라는 것은 10포인트 폰트에 12포인트 행간을 적용하라는 것이고 12/14는 12포인트 폰트에 14포인트 행간을 적용하라는 의미이다.

텍스트 한 행의 길이를 가리키는 말은 행장*(measure)*이다. 이것은 대부분 파이카(pica) 단위로 표시하는데 1파이카는 12포인트이다.

문서의 조판 상태는 특정 폰트, 특정 폰트 크기, 특정 행간, 특정 행장으로 지시할 수 있다. 보통 약부호로써 표현하는데 10포인트 폰트, 11포인트 행간, 20파이카 행장은 $10/11 \times 20$으로 나타낸다. $14/16 \times 22$은 14포인트 폰트 16포인트 행간, 22파이카 행장으로 판을 짜라는 것이다.

길이 측정 단위

타이포그라피와 인쇄업에서는 다양한 단위를 혼용한다. 기본 단위는 포인트인데, 표 [1]에 흔히 쓰이는 단위들을 보였다.

포인트와 파이카는 영어권 국가에서 전통적으로 쓰여온 인쇄 길이단위이다. 디도 포인트와 시세로는 유럽 국가에서 쓰여온 단위이다. 일본에서는 '규'(1/4 밀리미터)가 길이 단위로 사용된다.[10] 인치와 센티미터는 우리 모두에게 익숙하고 또 그래야 하는 단위이다.

[9] leading에 해당하는 '행송'이라는 표현이 있으나 여기서는 간단히 '행간'이라고만 하였다—역주.

[10] 일본어 '규'는 級의 일본식 발음이다. 이 일본식 발음으로부터 Q로 표시한다. 본래 사진식자에서 발전한 단위로서 1급은 1齒(1Q=1H)이다. 이 때의 齒란 사식기(CTS)의 드럼 톱니바퀴 하나의 간격을 의미하고 4齒가 1mm가 되도록 되어 있다. 급(규)은 길이 단위이고 치는 치송(齒送) 단위이다—역주.

포인트 체계는 소 피에르 푸르니에(Pierre Fournier le jeune)가 1737년에 처음 만들었는데 이 때 길이는 0.349mm 였다. 나중에 같은 세기에 프랑스와-앙브르와즈 디도(François-Ambroise Didot)가 독자적인 포인트 체계를 도입하였는데 이것은 0.3759mm 였다. 이 값이 지금까지 유럽에서 쓰이고 있다. 훨씬 후인 1886년에 미국 인쇄업 연합회(American Type Founders Association)에서 0.013837인치를 포인트의 표준 단위로 채택하였고 영국이 1898년에 이를 따랐다. 이 길이가 어느 정도 되는지 짐작이 가지 않는 사람을 위해서 말하자면 6파이카가 대략 1인치 정도가 된다.

큰 포인트(big point)는 약간 특이한 것으로 최근의 발명물이다. 이것은 PostScript[11]와 같은 페이지 기술 언어에서 계산을 좀더 빠르고 단순하게 할 수 있도록 고안된 것이다.

위의 모든 단위는 고정값이다. 어떤 폰트냐에 따라 달라지는 가변값 길이단위도 있다. *em*은 현재 폰트의 기본 설정 높이를 가리키는데 너비(width) 측정단위로 사용한다. *ex*는 현재 폰트의 'x' 문자 높이를 가리킨다. *quad*라는 용어를 이를테면 '1 quad 공백을 두고 시작한다'와 같은 문장에서 만날 수 있다. 이 길이는 em으로 정의된다. 대개 1 quad는 1em이다.

[11] 포스트스크립트는 Adobe Systems 사의 등록상표이다.

제 I 부

이론과 실제

제 1 장

책의 구성부분

1.1 서론

이 장에서는 책을 이루고 있는 여러 부분들에 대한 것, 각 부분의 순서, 전형적인 페이지 번호매김 방법에 대하여 다룬다.

1.2 앞부분(Front Matter)

책 한 권을 이루는 데는 세 부분이 있다. 앞부분(front matter 또는 preliminaries), 본문(main matter), 뒷부분(back matter) 또는 참조가 그것이다. 모양의 측면에서 주된 차이점은 앞부분의 면번호가 로만 번호로 붙고 절 구분에 숫자를 붙이지 않는 것이다. 본문과 뒷부분에서 면번호는 아라비아 숫자로 붙는다. 절 구분 표제에 숫자가 붙는 것은 본문에서이고 뒷부분에서는 숫자를 붙이지 않는다.

앞부분은 책 표지, 목차 등으로 이루어진다. 앞부분의 처음 몇 페이지에는 페이지 번호매김이 이루어지지 않는다. 따라서 면번호도 붙지 않는다. 나머지 부분은 번호매김을 하고 면번호도 붙이는데 로마자 숫자를 이용한다. 아래 보인 구성요소들이 모든 책에서 나타나는 것은 아니다.

첫 페이지는 홀수면으로 약표제지이다. 면번호는 붙이지 않는다. 이 페이지는 매우 단순하고 책의 표제만을 보여준다—부제, 저자 등 기타 정보는 표시하지 않는다. 이 페이지의 주요 목적은 주표지를 보호하는 것이다.

첫번째 짝수쪽 페이지는 약표제지의 뒷면이다. 만약 시리즈 중의 한 권이라면 시리즈 타이틀을 여기에 써주거나 기고자 목록을 싣거나 권두화(그림)을 넣거나 비워둔다. 시리즈 타이틀은 약표제지 페이지에 놓지 않고 저작권 표시 페이지에 둘 수도 있다.

표제지*(title page)*는 홀수쪽으로 책 전체의 완전한 표제, 저자, 편집자를 표시하고 이따금 출판사의 이름과 출판사 로고를 넣는 수도 있다.

표제지(도비라)는 간소하게 작성될 수도 있고 때로는 디자이너가 표현하고자 하는 의도에 따라 여러 가지 요소를 추가할 수도 있다. 어떤 경우든 이 페이지의 스타일은 책 본문에서 사용된 스타일에 대한 암시를 포함하여야 한다.

타이틀 페이지의 뒷면은 저작권 표지 페이지이다. 여기에는 저작권 고지, 출판 및 인쇄 연혁, 인쇄된 국가, ISBN 또는 CIP 정보 등이 포함된다. 이 페이지는 대개 기본 텍스트보다 조금 작은 폰트로 조판되는 것이 일반적이다.

저작권 페이지 다음에 헌사나 에피그라프를 홀수쪽에 둔다. 그 다음 짝수쪽 페이지는 비운다.

여기까지가 번호매김을 하지 않는 페이지들이다.

페이지 번호매김하는 페이지의 표제와 텍스트 배열 형식은 본문의 그것과 같아야 한다. 다만 표제에 번호는 붙지 않는다.

로마 숫자(i, ii)로 번호를 붙이는 첫번째 번호매김 페이지는 홀수쪽에 차례(ToC)라는 표제를 가지는 것이다. 책에 그림(삽화)나 표가 있다면 그림 차례(LoF) 또는 표 차례(LoT) 들이 목차 뒤에 빈 페이지 없이 따라온다. ToC는 주요 요소들에 해당하는 엔트리를 포함하고 있다. LoT가 있다면 이 자체가 ToC에서 언급된다. 본문의 장들은 당연히 열거되어야 하고 서문이나 참고문헌, 색인과 같은 요소들도 표시되어야 한다.

목차들 다음에 추천사(서언, foreword)가 빈 페이지 없이 올 수 있다. 추천사는 저자 자신이 아닌 대개 저명인사인 사람이 작성하는 것으로 글쓴이가 서명한다. 글쓴이의 서명은 마지막에 작은 대문자로 조판하는 것이 일반적이다.

저자서문(preface)은 보통 저자 자신이 작성하는 것으로 이 저작을 집필하게 된 동기나 본문에서는 하기 어려운 사적인 언급을 하는 부분이다. 저자서문은 추천사가 없으면 목차 뒤에, 있으면 추천사 바로 뒤에 빈 페이지 없이 놓인다.

저자서문에서 감사의 말을 하지 않고 별도로 사사(謝辭, acknowledgement)를 작성하는 경우도 있다. 이 때는 사사를 저자서문에 뒤이어 놓는다.

서론(introduction)을 본문의 일부로 하지 않는다면 그 다음에 나온다. 그리고 앞부분의 마지막 요소들로 약어 목록, 부호 목록, 사건 연대표, 가계도 기타 그 저작의 내용과 관련된 정보를 둔다.

표 [1.1]에 앞부분에 올 수 있는 요소들을 요약하였다.

추천사, 저자 서문, 서론 등의 표제는 서로 바꾸어 표기할 수 있다는 점을 알아두자. 어떤 책에서는 서론을 서문이라 하기도 하고 또 다른 책에서는 비슷한 구성요소에 다른 이름을 붙이기도 한다.

저작권 페이지

누구라도 표제, ToC, 서문 등에 대해서는 잘 알고 있겠지만 아마도 나와 마찬가지로 저작권 페이지에는 그다지 익숙하지 않은 사람도 있을 것이다. 이 부분은 출판사에서 작성

표 1.1: 앞부분

구성요소(Element)	Page	Paginated	Leaf
약표제지(Half-title page)	recto	no	1
권두화(Frontispiece) 또는 빈 페이지	verso	no	1
표제지(Title page)	recto	no	2
저작권 페이지(Copyright page)	verso	no	2
헌사(Dedication)	recto	no	3
빈 페이지(Blank)	verso	no	3
목차(Table of Contents)	recto	yes	3 or 4
그림 목차(List of Figures)	recto or verso	yes	3 or 4
표 목차(List of Tables)	recto or verso	yes	etc.
추천사(Foreword)	recto or verso	yes	etc.
저자서문(Preface)	recto or verso	yes	etc.
사사(Acknowledgements)	recto or verso	yes	etc.
서론(Introduction)	recto or verso	yes	etc.
약어표 기타(Abbreviations, etc)	recto or verso	yes	etc.

하는 경우가 많겠으나 어떤 저자들은 자의 또는 타의로 자신이 스스로 발행인이 되어야 하는 경우도 있는 것이다.

【memhangul】 한글 서적은 저작권을 '판권'이라 하고 저작권을 표시한 페이지를 판권지라 한다. 판권지를 앞부분에 두지 않고 책의 마지막에 두는 것이 종래의 출판 관행이었다. 최근 들어 양서와 같이 판권지를 앞부분에 두는 책이 늘어가고 있다. 이 번역에서는 판권 대신 저작권, 판권지 대신 저작권 페이지라는 표현을 썼다. ■

저작권 페이지의 핵심은 저작권 고지사항을 표시하는 것이다. 베른 조약에 따르면 출판된 저작물에 저작권 고지가 실려야만 저작권이 보호되는 것은 아니지만 그렇게 하는 것이 보다 안전하다고 믿고 저작권 고지를 포함하고 있다. 저작권 고지는 세 부분으로 구성되는데, 저작권 *Copyright*이라는 단어 또는 더 널리 쓰이는 © 기호가 맨 처음에 오고, 출판 연도, 그리고 저작권자 이름이 마지막에 온다. 저작권 기호는 국제저작권조약의 요구사항을 만족시켜야 한다. 이 조약에는 미국, 대부분의 유럽 국가들 및 많은 아시아 국가들이 가입하고 있다. 'All rights reserved'라는 표현이 부에노스 아이레스 조약에 따라 저작권 보호를 보증하기 위해 추가된다. 이 조약에는 대부분의 미주 국가들이 가입하고 있다. 전형적인 저작권 고지는 다음과 같이 표시된다.

© 2035 by Frederick Jones. All rights reserved.

같은 페이지의 어딘가, 대개 저작권 고지 근처에 발행인의 이름과 위치가 표시된다.

저작권 페이지에는 간행 연혁과 판(版, edition)[1], 간행일자가 표시되며, 가끔 인쇄된

[1]제2판은 초판보다 가치있는 것이어야 할 것이다. 그런 경우가 드물기는 하지만.

장소를 적시하기도 한다. 과거 내가 이해하지 못했던 것은 수수께끼같은 숫자의 나열이었다. 예를 들면 다음과 같은 것이다.

02 01 00 99 98 97 10 9 8 7 6 5

왼쪽 부분은 오른쪽에서 왼쪽으로 읽어야 하는데 초판이 간행된 해부터 시작해서 각 연도의 마지막 두 숫자를 적은 것이다. 오른쪽 부분은 역시 오른쪽에서 왼쪽으로 읽어야 하는데 새로 인쇄한 횟수(刷)를 보여준다. 각 부분의 가장 적은 숫자는 에디션 날짜와 현재 인쇄횟수를 나타낸다. 그러므로 위의 예는 1997년에 처음 출판된 책의 제5쇄라는 것을 의미하고 있다.

미국에서는 이 페이지에 의회 도서관 출판물 카탈로그(CIP) 데이터가 포함되는 경우가 있다. 이것은 의회 도서관이 발급하는 것으로 그 책의 키워드 일부를 제공한다.

또 이 페이지에는 ISBN(국제 표준 도서 번호)을 표시한다. 이것은 그 책의 고유 번호로서 예를 들어 ISBN 0-NNN-NNNNN-2 와 같은 형식을 취한다. 처음 0이라는 숫자는 이 책이 영어 사용권 국가에서 발행되었음을 의미한다. 그 다음 숫자들은 출판발행인, 그 다음은 그 발행인에 의해 부여된 책 번호를 의미하고, 마지막 숫자, 위에서 2라는 것은 확인을 위한 숫자이다.

CIP와 ISBN을 부여받는 것과 관련하여 더 많은 것을 알고 싶다면 직접 해보기 바란다.

1.3 본문

본문은 책의 핵심 부분이다.

본문의 모든 페이지는, 면번호를 찍지 않는 페이지가 있더라도 페이지 번호매김은 이루어진다. 면번호는 아라비아 숫자로 표시하는 것이 일반적이고 본문의 첫번째 오른쪽 페이지를 1로 시작한다.

본문은 기본적으로 장*(chapter)*으로 나누어진다. 어린이를 위한 짧은 이야기책이라면 그렇지 않겠지만. 장들이 모여서 편 또는 부*(part)*를 이룬다. 이것이 책의 가장 상위 구분 단위가 된다. 편과 장에는 번호를 붙이는 것이 일반적이다. 편의 번호는 책 전체를 통해서 일련번호가 붙어야 한다. 그리고 장의 번호는 편의 번호와 관계없이 일련번호가 붙는다.

편의 표제는 보통 홀수쪽에 둔다. 여기에는 편 타이틀과 편 번호가 온다. 장도 홀수쪽에서 시작하는 것이 일반적인데 이 경우에는 장의 본문이 장 표제와 같은 페이지에서 시작한다.

장은 절(section)로 구분된다. 절에는 번호가 붙을 수도 있고 그렇지 않을 수도 있다. 절의 번호붙임은 장이 바뀌면 새로 시작한다. 마찬가지로 절은 소절(subsection)로 구분될 수 있다. 특별한 학술 논문이 아니라면 장절 구획은 이 정도까지면 충분하다. 보통 장 내에서 절·소절 구획이 이루어질 때는 새로운 페이지를 시작하지 않는다.

편 또는 장의 타이틀 페이지는 면주를 붙이지 않는다. 그리고 새로운 장이 시작되기

전의 짝수쪽 면인 텍스트가 없는 백면에도 면주를 붙이지 않는다. 그 이외의 모든 페이지에는 면주를 붙여야 한다.

본문의 마지막 장은 결론 또는 맺음말이라는 제목이 붙는데, 여기서는 그 저작물 전체를 요약하고 다루지 못한 영역이나 향후 연구의 단서와 같은 것들을 기록한다.

번호붙는 부록이 추가되어야 한다면 논리적으로 볼 때 본문이 끝난 뒤에 와야 한다. 부록의 번호붙임은 숫자번호가 아니라 알파벳 '번호'를 사용하는 수가 많으므로, 첫번째 부록은 "부록 A", 두번째는 "부록 B" 등으로 표시한다.

에필로그 또는 후기(afterword)는 저자가 추가로 포함하는 비교적 짧은 언급이다. 이것은 그 앞의 장들보다 그 중요성에서 현저히 가볍게 취급되며 번호를 붙이지 않는 경우 뒷부분(back matter)에 넣는 경우가 많다.

1.4 뒷부분(back matter)

뒷부분은 필수적인 것은 아니지만 본문에 대해 보조적인 정보를 추가하고자 할 때 둔다.

번호붙지 않은 부록은 보통 뒷부분에 온다.

주석, 용어집, 기호표 또는 약어표 등을 앞부분에 두지 않고 뒷부분에 둘 수 있다. 기고자, 문헌목록, 색인 등도 올 수 있다. 이들은 보통 번호를 붙이지 않는다.

때에 따라 부록과 주석이 뒷부분이 아니라 각 장의 말미에 오는 경우도 있다.

뒷부분의 첫번째 요소는 홀수쪽에서 시작한다. 그러나 그 뒤로는 홀짝수쪽 어디에서도 시작할 수 있다.

옛날에는 간기(colophon)를 책의 제일 마지막에 두는 것이 관행이었다. 간기라는 것은 책의 제작과 디자인에 대한 사항을 적어두는 것인데 특히 어떤 폰트를 사용했는지를 항상 명시하였다.

1.5 접지와 재단, 판앉히기

전문적으로 서적을 인쇄할 때는 한 장의 큰 종이에 여러 페이지를 한꺼번에 인쇄한 다음 이것을 접지하여 작은 내지로 자른다. 이 내지들을 모아서[장합] 묶으면[철] 책이 되는 것이다. 접지 재단되지 않은 종이를 원지*(broadside)*라 한다. 원지를 한번 접지하면 *2*절지*(folio)*가 되어 두 장의 내지와 네 페이지가 만들어진다. 이것을 다시 접어서 자르면 두 장의 *4*절지*(quarto)*가 되고 네 장의 내지와 여덟 페이지가 나온다. 이것을 다시 반으로 접으면 네 장의 *8*절지*(octavo)*가 되어 여덟 장의 내지와 16 페이지가 만들어진다. 표 [1.2]를 보라.

표 [1.2]에서 Size 열은 원지 사이즈에 대해 재단되지 않은 크기를 보여준다. 책이 만들어지면 내지를 재단(trimming)하기 때문에 실제 크기는 디자이너의 의도에 맞추어 조금

표 1.2: 접지와 재단

Name	Folds	Size	Sheets	Leaves	Pages
원지 Broadside	0	$a \times b$	1	1	2
2절지 Folio	1	$b/2 \times a$	1	2	4
4절지 Quarto, *4to*	2	$a/2 \times b/2$	2	4	8
8절지 Octavo, *8vo*	3	$b/4 \times a/2$	4	8	16
16mo	4	$a/4 \times b/4$	8	16	32
32mo	5	$b/8 \times a/4$	16	32	64
64mo	6	$a/8 \times b/8$	32	64	128

작아진다. 대개 1/8 인치 내지 3밀리미터 정도가 상단 하단과 내지의 바깥쪽에서 잘려나간다.

접장을 만드는 접지 방법은 이 외에 다른 것도 있다. 예를 들면 *6절(sexto)*이라는 것은 각 장을 3분으로 접은 다음 다시 반으로 접은 것으로 세 장의 6절지가 나오고 여섯 장의 내지와 열두 페이지가 만들어진다.

각각의 접장(signature)을 순서대로 포개어서[장합] 접장의 접이 부분을 매기(철) 하면 책이 만들어진다. 접장을 모으고 표지와 면지(endpaper), 책등(spine)을 붙이면 전 과정이 완성된다.

출판인들은 최종적으로 조판된 책이 어떤 접장 총수 범위 안에 맞아들어가기를 원한다. 접장 수를 맞추기 위해서 빈 페이지를 넣는 것을 되도록 적게 하고 싶어한다. 조판어림이란 어떤 사이즈의 활자로 주어진 텍스트가 몇 행이나 되는지, 그래서 결과적으로 몇 페이지가 필요한지를 결정하는 과정을 가리킨다.

조판어림을 위해서는 한 행에 몇 글자가 들어가는지, 전체 텍스트에 몇 글자가 있는지를 알아야 한다. 조판어림에서 '글자'란 문자 숫자 문장부호를 모두 가리키는 말이다. 행당 글자수는 비교적 쉽게 알 수 있다. 조견표를 보거나 직접 계산해보면 된다. 자세한 것은 §2.4를 참고하라. 텍스트의 문자수는 좀 골치아프다. 특히 초고가 아직 완성된 상태가 아니라면 더 그렇다. 영문 텍스트의 경우 한 단어는 평균 다섯 글자 정도라고 하므로 여기에 스페이스 하나를 더하면 여섯 '글자'가 된다고 보고 대략 개산하는 방법을 많이 쓰는데 학술 문서의 경우에는 대개 이 값보다 조금 더 긴 단어가 사용될 것이라 볼 수 있을 것이다.

단어 수를 알아보려면 가장 쉬운 방법은 초고의 대표적인 부분을 타이핑한 다음에 그 부분의 단어수를 헤아려서 그것을 전체 텍스트에서 차지하는 비율로 나누는 것이다. 예를 들어 전체의 1/20을 타이프쳤다면 그 부분의 단어수를 1/20으로 나누면 되는데 이것은 결국 20을 곱하는 것과 같다. 필요한 페이지수를 가늠하려면 장 표제, 삽화 등에 대해서도 고려해야 한다.

만약 전체가 3장의 접장과 2페이지로 이루어져 있다면 이것을 세 장으로 하거나 네

장의 접장에서 한두 페이지가 빠지는 정도로 맞추는 것이 편리할 것이다. 텍스트를 확장하거나 축약하거나 폰트를 바꾸거나 페이지당 행수를 변경하면 된다.

내가 학술 저널의 편집을 맡고 있을 때 저자들에게 단어수의 상한선을 정해주었다. 그렇게 했던 이유는 우리가 실제 단어수에 관심이 있었기 때문이 아니고 오히려 개략적인 또는 개연적인 한계를 추정함으로써 각 논문마다 할당될 페이지수를 알고 싶었기 때문이다. 우리는 *8*절판 접장을 사용했고 빈 페이지를 넣지 않았다. 이러한 우리의 경험은 아마도 모든 출판인에게 공통된 것이 아닐까 한다. 즉 단어수가 중요한 것이 아니라 페이지 수가 중요한 것이다.

몇몇 특별한 경우에 책의 본문에 추가 페이지가 '별지'로 들어갈 때가 있다. 대부분 비용 문제 때문에 삽화를 별도의 페이지로 인쇄하여 책 사이에 끼워넣는 경우에 사용된다. 또다른 경우로 접어넣은 쪽이 있는데 이것은 예컨대 아주 큰 지도 같은 것이나 삽화의 확대 그림을 넣기 위해서 쓴다. 별지 삽입된 페이지는 책 안의 적절한 위치에 묶여서 제책이 되지만 페이지번호는 붙을 수도 있고 붙지 않을 수도 있다. 별지삽입의 경우에는 삽화 목록이 '52에서 53페이지까지'와 같은 방식으로 지시될 것이다.

제 2 장

페이지

책을 쓴다는 것은 자신이 읽기 위해서가 아니라 다른 사람에게 읽히기 위한 것이다. 그러므로 저자의 원고를 초고 상태 그대로 내놓을 수는 없다. 또한 저자들은 출판된 자신의 저작이 독자를 사로잡기를 원할 것이다. 여기에는 두 가지 측면이 있다. 보다 중요한 것은 내용 자체이다. 저자의 생각이 흥미롭게 표현되었는가 하는 것. 만약 내용 자체가 지겨운 것이라면 한번 흘낏 보고 심각하게 읽어보고자 했다 하더라도 그 책을 끝까지 파고 들어서 고통받는 것보다 훨씬 재미난 일이 많을 텐데 그런 일을 왜 하겠는가. 또다른 측면은 그 내용을 보여주는 방식이다. 이것을 타이포그라피라고 한다. 그리고 이것이 이 장의 주제이다.

좋은 타이포그라피의 핵심은 한눈에 알아볼 수 있는 것이 아니다. 두번세번도 마찬가지다. 그것은 훈련된 눈이 아니라면 알아보기 힘든 것이다. 만약에 책 전체를 훑어보고난 다음 맨처음 보인 반응이 그 레이아웃에 감탄하는 것이라면, 그것도 디자인이라 한다면 최악의 디자인이다. 좋은 타이포그라피는 미묘한 것으로 눈에 확 띄는 그런 것이 아니다.

데스크탑 출판이 보편화되면서 많은 저자들은 자신의 책을 스스로 디자인하려 한다. 얼핏 보기에 별것아닌 것 같다. 폰트는 많겠다, 그 중에서 몇 개를 골라내서 제목줄에는 이 폰트 본문에는 저 폰트 캡션에는 또다른 폰트를 쓰고, 여백설정해서 본문 폭을 정하면 그만 아닌가.

그러나, 글쓰기가 숙련을 요하는 기술인 것과 마찬가지로 타이포그라피 역시 숙련과 경험을 요하는 분야이다. 책 한 권의 좋은 디자인에는 수백년의 경험이 축적되어 있다. 이것은 가볍게 무시할 수 있는 것이 아니다. 스스로 디자인까지 하는 자가저작자들은 어렵사리 얻은 교훈을 알지 못하고 자신들이 하는 일이 그와 정반대라는 것조차도 알지 못하는 경우가 많다. 전문가도 규칙을 깨뜨릴 수 있다. 그러나 그들은 그 규칙이 무엇인지 알고 있으며 자신이 그것을 깨뜨리는 데 대해 합당한 이유를 가지고 있다. 자가저작자는 타이포그라피 규칙을 일상적으로 깨뜨리지만 규칙이 무엇인지 알지도 못하고 있는 것이다.

저자는 메시지를 만들고 타이포그라퍼는 매체를 만든다. 마샬 맥루한의 의견과는 반대로 매체라는 것은 메시지가 0닌 것이다. 타이포그라퍼가 하는 일은 메시지와 청중 사이에 끼어들려는 것이 아니라 가만히 독자의 흥미를 유발하고 메시지에 빠져들도록 만드는 것이다. 어떤 책의 디자인과 레이아웃이 '나 좀 봐줘!' 하고 소리치고 있다면 이것은 디자인이 아니라 광고이며 디자이너의 작업으로서는 이보다 더 나쁠 수 없는 것이다.

2.1 책의 형태

책의 크기와 모양은 다양하다. 그러나 수 세기에 걸쳐 보기 좋고 편리한 형태가 발전해왔다. 그래서 약간의 예외를 별도로 한다면 대개 책은 사각형 형태가 되었다. 예외는 거의 어린이용 책이다. 다만 내가 가진 책 중에 Fritz Spiegl이 편집하고 Pan Books에서 출판한 *A Small Book of Grave Humour*라는 것이 있기는 한데 이것은 묘비 모양을 하고 있는 묘지명 모음집이다. 책을 닫은 상태에서는 보통 세로가 가로길이보다 길다. 미학적인 면을 논외로 하면 이 형태의 책이 가로가 긴 책보다 손에 들고 보기 편리하다.

디자이너가 책의 크기를 결정하는 것은 맘대로 해도 될 성싶다. 그러나 경제적 이유 때문에 반드시 그렇지가 않다. 타이포그라피 디자인은 용지의 표준산업규격에 따를 수밖에 없다. 12×8 인치 크기의 종이를 쓰면 미국 표준인 11×8 1/2 인치 letter 용지보다 훨씬 비싸게 먹힌다. 마찬가지로 업무용 봉투 규격인 4 1/8×9 1/2 인치 용지도 표준 규격이다. 우편송달을 위한 브로셔를 작성하려 할 때 이것은 접은 최소 크기가 봉투에 들어갈 수 있도록 디자인해야 할 것이다. 그래서 5×10 인치 크기 브로셔를 만들면 보기 좋으냐 나쁘냐를 떠나서 불편할 수밖에 없다.

지금까지 출판된 책의 종횡비는 대단히 다양하다. 종횡비란 가로 대 세로 비를 말한다. 그러나 수세기에 걸쳐서 나라와 문명을 막론하고 반복해서 출현하는 종횡비가 있다. 이것은 어떤 종류의 비례가 내재적으로 사람의 눈에 아름답게 보이기 때문이다. 이러한 아름다운 비례는 자연에서도—물리, 생물, 화학 현상이나 건축물에서도—발견된다.

아름다운 비례의 보기로서 일본 목판화에서 발견되는 것이 있다. 호소에 사이즈(2 : 1)는 2배 직사각형이다. 오반 사이즈는 3 : 2이고 주반은 11 : 8이며 코반 사이즈는 $\sqrt{2} : 1$이다. 이 목판화가 책으로 묶어진 일도 가끔 있기는 하지만 판화 한 장이 하나의 작품인 경우가 많았다. 마찬가지로 인도 회화에서 최소한 16 내지 18세기 동안 1.701 : 1에서 13 : 9의 비율이 나타나기도 하였다. 대체로 3 : 2를 전후한 종횡비에 해당한다.

중세 유럽에서는 페이지 비례가 일반적으로 1.25 : 1에서 1.5 : 1에 이르고 있다. 종이 크기는 4 : 3 (1.33 : 1)이나 3 : 2 (1.5 : 1)의 비율인데 이것을 차례로 접어서 그러한 비율이 나오게 되었다. 예를 들어 종이가 처음에 60×40 (즉 3 : 2)의 비율이었다면 첫번째로 접어서 만들어지는 크기는 30×40 (즉 3 : 4)이 된다. 다시 한번 더 접으면 30×20, 즉 3 : 2가 된다. 흥미로운 것은 아무리 종이가 크고 얇아도 여섯 번 이상 접는 것은 불가능하다는 것이다. 르네상스 시대의 출판업자들은 세로가 더 긴 책을 만들려고 하였다. 이 때의

종횡비는 1.87 : 1에 이르렀다. 요즘 책은 오히려 중세 시대의 비율로 되돌아가는 경향이 있다.

ISO 표준 규격의 종이 종횡비는 $\sqrt{2} : 1$ $(1.414 : 1)$이다. 이 비례는 중세의 것과 비슷한 것이지만 여러 번 접더라도 매번 같은 비율이 되게 한다. A0 원지 $(1189 \times 841$ mm)는 한번 접어서 A1 $(594 \times 841$ mm) 사이즈가 되고 다시 접으면 A2 (420×594)가, 다시 A3, A4 $(210 \times 297$ mm), A5 사이즈가 된다.

어떤 것이 분명히 좀더 나은 것은 있을지 몰라도 완벽한 페이지 종횡비란 있을 수 없는 것이다. 일반적인 서적에서 출판업자와 독자들이 모두 선호하는 비율은 대체로 $9 : 5$ $(1.8 : 1)$에서 $5 : 4$ $(1.25 : 1)$ 범위에 걸쳐 있다. 그림 (2.1)에 몇 가지 예를 보였다. $\sqrt{2} : 1$ $(1.414 : 1)$보다 더 넓은 페이지는 주로 표나 마진 노트를 위해 여분의 폭이 필요한 문서의 경우이거나 다단 편집이 적합한 경우에 쓰인다.

삽화가 중요한 책에서 그림의 형태는 페이지 비례의 선택에도 영향을 미친다. 페이지 크기는 삽화들의 평균적인 크기보다 좀더 커야 한다. 삽화를 설명하는 캡션을 넣기 위해서 세로 길이를 더 확보해야 한다. $\pi : e$ $(1.156 : 1)$ 비율은 정사각형에 비해 약간 긴 모양인데 정사각형 삽화를 넣기에 좋다.[1] $e : \pi$ $(0.864 : 1)$ 비율은 4×5 카메라로 찍은 가로가 긴 사진첩에 유용하다. 35mm 카메라(2 : 3 비율을 반대로 한 사이즈)로 찍은 사진의 경우에는 $0.83 : 1$ 페이지에 잘 맞는다.

2.2 황금비와 피보나치 수열

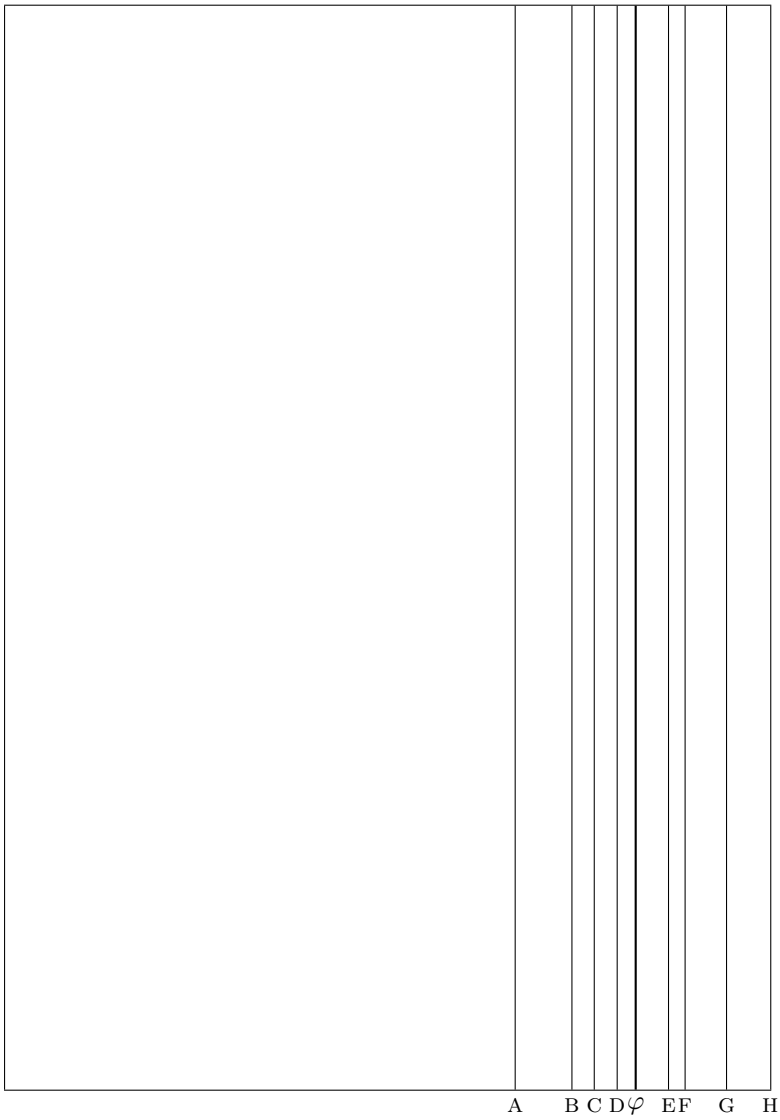
타이포그라퍼에게 필요한 수학 능력이란 평범한 10대 소년소녀가 할 수 있는 정도—기본적인 간단한 산수를 넘지 않는다. 흥미로울 수도 있는 몇 가지 수학적인 배경을 검토해보려 하므로 원한다면 이 절을 건너뛰어도 좋다.

고대 그리스 시대 또는 그보다 훨씬 전부터, 그리스 글자 φ (phi)로 표시되는 황금 분할은 특히 조화로운 비례라고 알려져 왔다. 이 비례가 타이포그라피에서도 적용되고 있는 것은 놀라운 일이 아닐 것이다.

그리스인들은 기하학에 관심이 많았다(유클리드를 생각해보라). 그들은 직선을 어떤 방법으로 비대칭 분할하면 특히 미학적인 자질을 가진 비례를 찾아낼 수 있다는 사실을 발견했다. 선분 l이 주어져 있을 때 이것을 a와 b의 두 부분으로 나누되, a보다 b가 더 길도록 나누는 것으로 하자. 이 때 긴 부분선분과 짧은 부분선분의 비(b/a)가 선분 전체와 긴 부분선분의 길이의 비(l/b)와 같아지도록 하라는 것이다. 다시 말해보면 φ로 표시되는 황금 분할이란 큰쪽과 작은쪽의 비가 그 둘을 합친 길이의 큰쪽에 대한 비와 같도록 하는 것이다. 두 요소를 각각 a와 b라 하고 $a < b$라면 황금 분할은 다음과 같이 표시된다.

$$\varphi = \frac{b}{a} = \frac{a+b}{b} = (1+\sqrt{5})/2 \tag{2.1}$$

[1] e나 π는 모두 잘 알려진 수들이다. e $(= 2.718\ldots)$는 자연로그의 밑이고 π $(= 3.141\ldots)$는 원주율이다.



A 2 : 1
B 9 : 5
C 1.732 : 1 ($\sqrt{3} : 1$)
D 5 : 3
φ 1.618 : 1 ($\varphi : 1$)
E 1.538 : 1
F 3 : 2
G 1.414 : 1 ($\sqrt{2} : 1$)
H 4 : 3

그림 2.1: 페이지 비례의 예

황금 분할은 여러 가지 이름으로 불린다. 유클리드는 이것을 '극단적이고 중용적인 비례(extreme and mean ratio'라고 하였고 르네상스 시기에는 '신성한 비례'라고까지 불렸다. 현재는 '황금 분할' 또는 '황금비'라고 부르고 있다. φ 기호는 자신의 작품에 황금비를 자주 활용한 그리스 예술가 피디아스(C5th BC)의 이름에서 온 것이다. 황금비례로 이루어지는 직사각형을 '황금 직사각형'이라 한다. 아테네 아크로폴리스의 파르테논 신전의 전면은 황금 직사각형이다. 이 직사각형은 그리스 건축 곳곳에서 나타난다. 피타고라스 학파의 상징인 별 모양의 펜타그램에서도 각 선분은 황금비율로 분할된다.

φ의 십진표기 근사값은 1.61803이다. 이 숫자는 몇 가지 특이한 성질을 가지고 있는데, φ에 1을 더하면 그 제곱값을 얻게 되고 1을 빼주면 그 역수가 된다.

$$\varphi + 1 = \varphi^2 \tag{2.2}$$

$$\varphi - 1 = 1/\varphi \tag{2.3}$$

이 숫자는 또한 아주 간단한 연분수로도 정의할 수 있다.

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots}}}} \tag{2.4}$$

Leonardo Fibonacci라고 알려진 Leonardo Pisano는 1202년에 *Liber Abbaci*[2]라는 책을 썼다. 이 책의 주제 중에서 그가 관심을 가졌던 것 하나는 인구 증가에 대한 것이었다. 이 책에서는 다음과 같은 계산을 수행하고 있다.

> 토끼 한 쌍이 일년에 낳을 수 있는 토끼는 몇 쌍이나 될까? 토끼 한 쌍이 한 달마다 새로운 한 쌍을 생산한다고 하자. 토끼는 한 달이면 생식이 가능해진다. 그리고 일년동안 어떤 토끼도 죽지 않는다고 가정한다.

한 달이 지나면 두 쌍이 된다. 다음 달 마지막에는 첫번째 쌍이 또다른 한 쌍을 만들어낼 것이므로 세 쌍이 된다. 그 다음 달 마지막날에는 원래의 한 쌍이 세번째 쌍을 낳을 것이고 처음 태어난 한 쌍도 역시 한 쌍을 낳을 것이므로 모두 합해서 다섯 쌍이 된다. 그 다음 달에는.... 이 계산을, 토끼처럼, 지치지도 않고 계속하게 되면 다음의 수열을 얻는다.[3]

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \ldots$$

처음 두 항을 건너뛴 다음부터 각 항은 그 이전 두 항의 합으로 이루어진다. 이 수열의

[2] 계산의 책

[3] 이 수열의 첫번째 숫자는 처음에 생식가능한 토끼와 그 새끼가 몇 쌍 있다고 가정하느냐에 달려 있다.

인접한 두 항의 비율은 φ ($= 1.618\ldots$) 값을 사이에 두고 진동하면서 점점 그 값에 수렴해간다.

$$\begin{aligned} 8/5 &= 1.6 \\ 13/8 &= 1.625 \\ 21/13 &= 1.615 \\ 34/21 &= 1.619 \\ 55/34 &= 1.6176 \\ 89/55 &= 1.6182 \end{aligned}$$

φ 값과 피보나치 수열 사이에는 수학적으로 대단히 흥미로운 또하나의 (적어도 나에게는) 놀라운 관계가 있다. 피보나치 수 F_n을 다음과 같이 정의하자.

$$F_0 = 0; \quad F_1 = 1; \quad F_{n+2} = F_{n+1} + F_n, \quad n \geq 0. \tag{2.5}$$

그러면 다음 식이 성립한다.

$$F_n = \frac{1}{\sqrt{5}}(\varphi^n - (-\varphi)^{-n}) \tag{2.6}$$

피보나치 수열과 황금 비례는 자연에서도 발견할 수 있다. 해바라기 씨의 배열, 솔방울의 표면 패턴, 줄기 주변에서 잎이 나는 위치 등이 모두 피보나치 패턴을 보여준다 ([CG96]을 참고하라). 마틴 가드너 [Gar66]에 의하면 사람의 키와 배꼽 높이 사이의 평균적인 비율이 1.618+(φ에 가깝다면 가까운 값)임을 주장하는 연구가 있다고 한다.

2.3 펼침면

편집영역은 본문 문자들이 배열되는 페이지의 한 부분이다. 페이지 형태를 결정하는 데 유용한 비율은 편집 영역의 종횡비를 결정하는 데도 역시 유용하다. 이것이 페이지의 종횡비가 편집영역의 종횡비와 동일해야 한다는 말은 아니다. 예를 들어 정사각형 페이지에 정사각형 편집영역을 놓는 것은 바보같은 짓이다.

맨처음에 우리가 배운 것은 각 행을 따라가며 가로로 읽는 법이었다. 숙련도가 증가해가면서 우리는 가로보다는 세로로 대강 훑어보는 방법에 익숙해졌다. 텍스트가 세로로 길게 놓여 있고 가로가 너무 길지 않으면 이렇게 하기가 더 용이해진다.

책의 페이지는 대체로 몇 가지 요소를 가지고 있다. 이 가운데 중심적인 것은 편집영역이다. 그외에도 면번호(folio)(즉 페이지 번호), 페이지 상단 면주와 하단 면주(여기에는 장 제목이나 책 표제가 실린다), 그리고 여백주석(marginalia) 및 각주와 같은 요소들이

있다. 이 나머지 요소들은 그것이 책의 내용에 비추어보아 필수적인 것이라 하더라도 편집영역보다는 시각적으로 덜 중요하게 취급된다. 그러나 사소한 장식 하나가 훌륭한 디자인을 망가뜨리기에는 충분하다.

편집영역을 페이지 위에 위치시키는 것이 가장 중요한 관심사이다. 편집영역의 위치를 결정하는 것은 또한 여백의 범위를 결정하는 결과를 가져온다. 페이지 디자인이란 페이지의 종횡비를 편집영역과 마진의 비율과 균형을 맞추어서 흥미롭고도 조화로운 구성을 만들어내는 작업이다. 표제지를 제외하면 단 한 페이지는 디자인의 대상이 될 수 없다. 디자인은 양면 맞쪽의 두 페이지를 대상으로 하는데 이것은 책을 펼쳤을 때 보이는 모양이다. 즉 왼쪽 페이지와 오른쪽 페이지는 하나의 디자인 단위로 간주된다. 좀더 테크니컬하게 말하자면 디자인은 양면 펼침면을 대상으로 하는 것이다.

표 [2.1]에는 페이지 디자인의 몇 가지 보기가 나와 있다. 뒤로 갈수록 세로 대 가로비가 커지는 순서로 배열되어 있다. 이 표에서 페이지의 종횡비를 나타내기 위해 한 개의 숫자를 사용했는데 예컨대 1.5라는 것은 $1.5:1$을 나타내는 것이고 12/7이라는 것은 $12:7$을 의미한다. 앞으로도 이 표기법을 사용하겠다. 이 표에 사용된 부호의 의미는 다음과 같다.

비율 :

P = 페이지 종횡비 = h/w
T = 편집영역 종횡비 = d/m

페이지 사이즈 :

w = 페이지 가로길이(폭)
h = 페이지 세로길이(높이)

편집영역 :

m = 주편집영역의 행장(폭)
d = 깊이(면주 영역 제외)

여백 :

s = 등쪽(안쪽) 마진
t = 위쪽 마진
e = 배쪽(바깥쪽) 마진
f = 아래쪽 마진
g = 단간 구획폭(다단 페이지에서)

그림 (2.2)에서 2.17까지 이 디자인의 모양을 보였다. 각각 양면 펼침면으로 나타내었고 페이지 폭은 비례를 시각적으로 비교해볼 수 있도록 고정시켜 나타내었다. 여기서 보인

표 2.1: 몇 가지 페이지 디자인

P	T	Margins & Columns					Figure
		s	t	e	f	g	
$\sqrt{3}$	2	$w/13$	$8s/5$	$16s/5$	$16s/5$		2.2 left
$\sqrt{3}$	e/φ	$w/10$	$2s$	$2s$	$3s$		2.2 right
12/7	1.701	$w/7$	$8s/5$	$8s/5$	$14s/5$		2.3 left
e/φ	7/4	$w/10$	$5s/4$	$5s/3$	$11s/8$		2.3 right
φ	1.866	$w/9$	s	$2s$	$7s/3$		2.4 left
φ	φ	$w/12$	$2s$	$5s/2$	$4s$		2.4 right
8/5	1.634	$2w/15$	$7s/5$	$9s/5$	$13s/5$		2.5 left
19/12	7/4	$2w/15$	s	$9s/8$	$11s/8$		2.5 right
19/12	$\sqrt{3}$	$w/7$	s	$5s/4$	$1.84s$		2.6 left
19/12	8/5	$w/12$	$7s/5$	$8s/5$	$2s$		2.6 right
$\pi/2$	9/5	$w/9$	$3s/2$	$5s/2$	$3s$		2.7 left
$e/\sqrt{3}$	1.71	$w/10$	$11s/8$	$24s/11$	$8s/3$		2.7 right
1.553	1.658	$w/11$	φs	φs	φs		2.8 left
1.538	$\sqrt{7}$	$w/10$	s	$23s/6$	$3s/2$		2.8 right
3/2	2	$w/5$	$s/2$	s	s		2.9 left
3/2	1.701	$w/9$	s	$2s$	$7s/3$		2.9 right
3/2	$\pi/2$	$w/13$	$2s$	$10s/3$	$30s/7$		2.10 left
3/2	3/2	$w/9$	$3s/2$	$2s$	$3s$		2.10 right
3/2	1.68	$w/23$	$2s$	$5s$	$2s$		2.11 left
3/2	3/2	$w/10$	$2s$	$5s/2$	$2.85s$		2.11 right
1.48	1.376	$w/12$	$7s/4$	$2s$	$7s/2$		2.12 left
13/9	$\sqrt{2}$	$w/30$	$2s$	$9s/2$	$4s$	$s/2$	2.12 right
$\sqrt{2}$	φ	$w/9$	s	$2s$	$2s$		2.13 left
$\sqrt{2}$	φ	$w/8$	s	$5s/3$	$5s/3$		2.13 right
7/5	1.641	$w/7$	s	$8s/5$	$8s/5$		2.14 left
1.294	φ	$0.176w$	$1.03s$	$1.685s$	$13s/9$		2.14 right
1.294	13/9	$w/12$	s	$2s$	$10s/7$	$s/2$	2.15 left
9/7	19/9	$2w/5$	$5s/8$	$5s/8$	$5s/6$		2.15 right
5/4	13/11	$w/10$	$3s/2$	$2s$	$8s/3$		2.16 left
7/6	17/15	$w/13$	s	s	$7s/5$	.382	2.16 right
e/π	0.951	$w/9$	s	$2s$	$3s/2$		2.17 left
5/7	2/3	$w/9$	$s/2$	$2s/3$	s	$s/3$	2.17 right

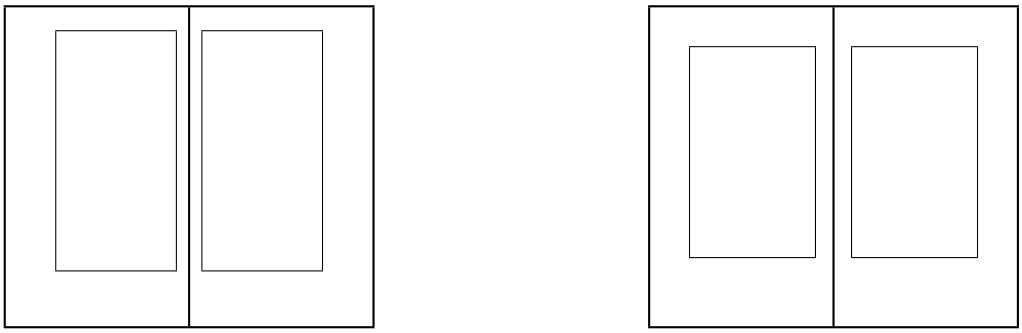


그림 2.2: 양면 펼침면: (왼쪽) 캐나다 1992. (오른쪽) 영국, 1970.

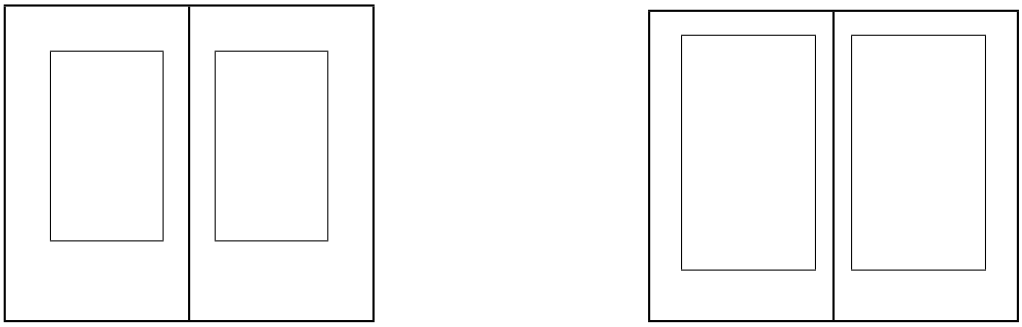


그림 2.3: 양면 펼침면: (왼쪽) 미국, 1909. (오른쪽) 영국, 1964.

것은 절대 크기가 아니라 상대적인 비례만을 표시한 것이다. 페이지와 편집영역만을 나타냈는데 그것은 다이어그램에 상단 면주, 하단 면주, 면번호 등을 표시하면 복잡해지기 때문이다.

그림 (2.2)에 보인 것은 현대 서적 두 권이다. 왼쪽의 것은 Robert Bringhurst의 *The Elements of Typographical Style* (Hartley & Marks, 1992)의 레이아웃으로 Bringhurst 자신이 디자인한 것이다. 본문 글꼴은 Minion체 12pt, 행장은 21pc이다. 캡션은 Syntax체를 사용하였다. 원 크기는 227×132mm이다. 타이포그라피에 관심이 있다면 꼭 읽어야 할 책으로 추천한다. 오른쪽 레이아웃은 Folio Society의 1970년판 마키아벨리(Niccolò Machiavelli)의 군주론(*The Prince*)이다. 원 크기는 216 × 125mm이고 12/13 × 22 Centaur체로 조판되었다.

그림 (2.3)의 왼쪽은 Wilfred T. Grenfell의 소책자 *Adrift on an Ice-Pan*의 레이아웃이다. 이 책은 Boston의 Riverside 출판사에서 1909년에 간행하였다. 텍스트는 16 포인트 행간에 16 파이카 행장으로 조판되었다. 행간은 넓고 행장은 좁아서 매우 개방적인 모습을 보여준다. 원본은 184 × 107mm 사이즈이다. 오늘쪽의 것은 Folio Society의 다른 책, Jerome K. Jerome의 *Three Men in a Boat*로서 1964년에 출간된 것이다. 원래 크기는

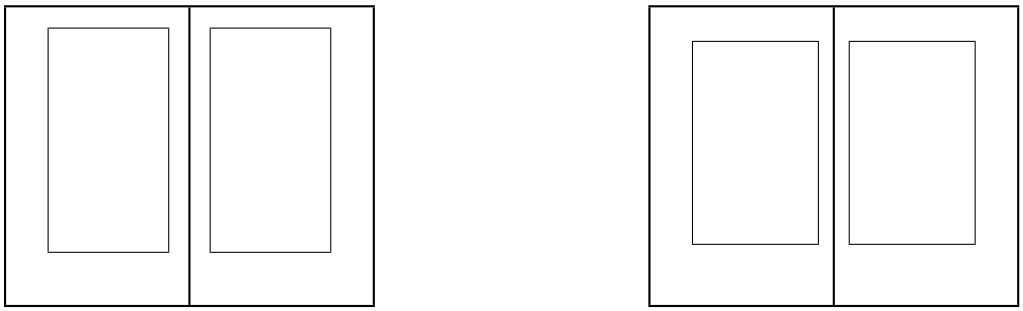


그림 2.4: 양면 펼침면: (왼쪽) 프랑스, 1559. (오른쪽) 캐나다, 1995.

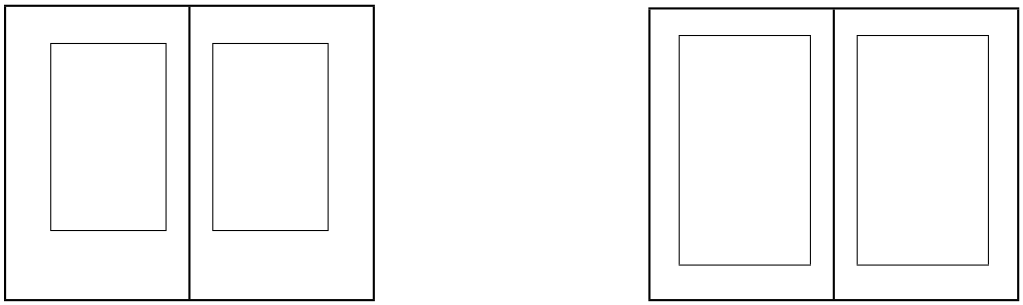


그림 2.5: 양면 펼침면: (왼쪽) 미국, 1949. (오른쪽) 미국, 1990.

215 × 128mm이고 Ehrhardt체 11/12 × 22로 조판되었다.

파리의 출판업자 Jean de Tourmes는 Jean Froissart의 *Histoire et Chronique*를 1559년에 출간하였다. 이것은 본문을 로만체로 하고 본문 사이즈의 약 80% 에 이르는 이탤릭 사이드노트를 가진 역사책이다. 그림 (2.4)의 왼쪽에 보인 레이아웃이 그것이다. 그림에서는 보이지 않지만 본문과 사이드 노트 칼럼 사이에 구분자(gutter)가 매우 협소하지만 폰트와 폰트 사이즈의 변화로 혼동 없이 읽을 수 있게 하고 있다. Hartley & Marks 사의 또다른 타이포그라피 서적인 Geoffrey Dowding의 *Finer Points in the Spacing & Arrangement of Type*이 그림 (2.4)의 오른쪽이다. Ehrhardt체 10.5/14 × 23으로 231 × 143mm 페이지에 조판되었다.

Bruce Rogers(1870–1957)는 자신의 Centaur 폰트를 디자인하게 된 계기를 1949년 10월에 자신의 스튜디오에서 사적으로 간행한 *Centaur Types*라는 책에서 기술하고 있다. 당연히 Centaur 폰트로 조판한 이 책의 레이아웃을 그림 (2.5)의 왼쪽 그림이 나타내고 있다. Centaur는 Nicolas Jenson이 1470년에 간행된 *Eusebius*에서 사용한 활자에 기초하고 있는 길쭉한 세리프 글꼴이다. *Centaur Types*에는 Centaur 폰트 이외에 다른 글꼴도 시험하고 있으며 도안의 정확한 크기를 싣고 있다. 이 책은 240×150mm 용지에 14/16×22로 조판하였다. 그림 (2.5)의 오른쪽 그림은 글꼴에 관한 또다른 책 한 권의 레이아웃이다.

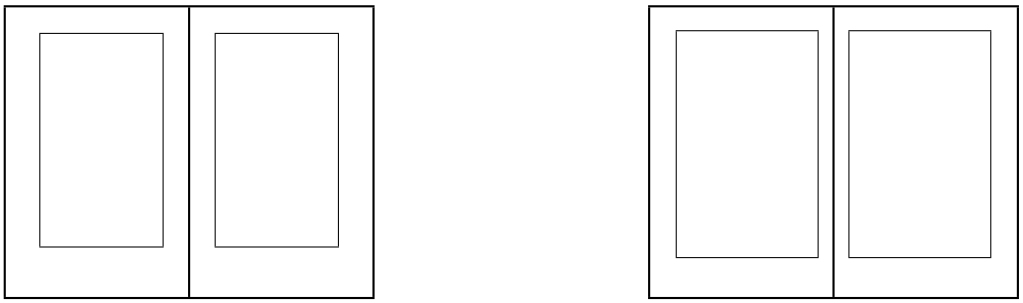


그림 2.6: 양면 펼침면: (왼쪽) 영국 1908. (오른쪽) 미국 1993.

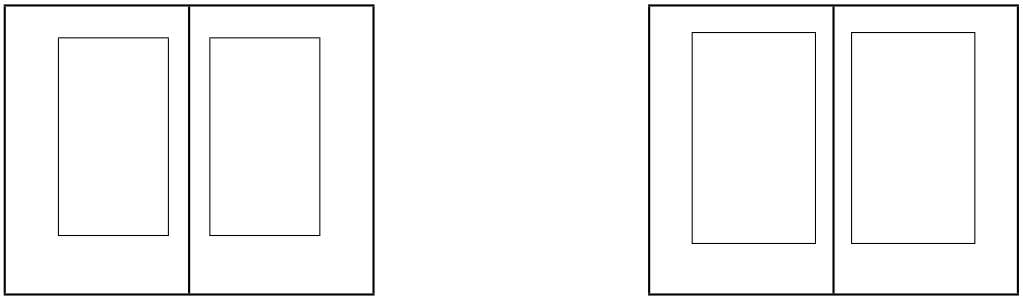


그림 2.7: 양면 펼침면: (왼쪽) 미국, 1931. (오른쪽) 영국, 1968.

David R. Godine이 1990년에 출간한 Alexander Lawson의 *The Anatomy of a Typeface*가 그 책이다. Galliard 폰트를 사용하여 13 포인트 행간과 24 파이카 행장의 227.5 × 150mm 페이지로 만들어졌다.

그림 (2.6)의 왼쪽에 보인 것은 F. M. Cornford의 *Microcosmographica Academia*이다. 제목은 라틴어를 썼지만 실제로는 영어로 쓰여진 책이고 London의 Bowes & Bowes 사가 1908년에 출판한 것이다. 이 책은 19세기의 전환기에 캠브리지 대학에서 벌어지던 학내 정치행태를(20세기라도 달라지지는 않았겠지만) 냉소적인 유머로 그리고 있는 책이다. 행간은 14 포인트이고 행장은 22 파이카이다. 원본 페이지 크기는 216 × 136mm이다. 그림의 오른쪽은 역시 제목이 범상치않은 David R. Dodine이 1993년에 출판한 Richard A. Firmage의 *The Alphabet Abcedarium*이다. 아도비 가라몬드 폰트로 조판하였는데 행장은 27 파이카이고 행간은 14 포인트이다. 원본 페이지 사이즈는 227.5 × 150mm이다. 이 책은 라틴 알파벳 문자 각각의 역사를 기술하고 있다. 특이한 점은 페이지마다 깊은 footer를 확보하여 기술하고 있는 글자의 여러 가지 문자꼴을 보여주고 있다는 점이다.

미국인 W. A. Dwiggins는 무엇보다 북 디자이너였다. 그림 (2.7)은 그가 디자인한 Random House 1931년간 H. G. Wells의 *The Time Machine*의 레이아웃을 보여준다. 페이지 사이즈는 231 × 147mm이다. 오른쪽 그림은 캠브리지 대학 출판부에서 1968년에 간행된

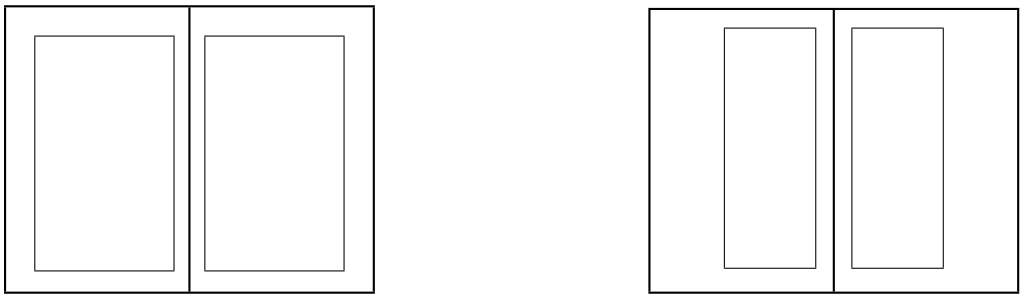


그림 2.8: 양면 펼침면: (왼쪽) 미국, 1994. (오른쪽) 영국, 1988.

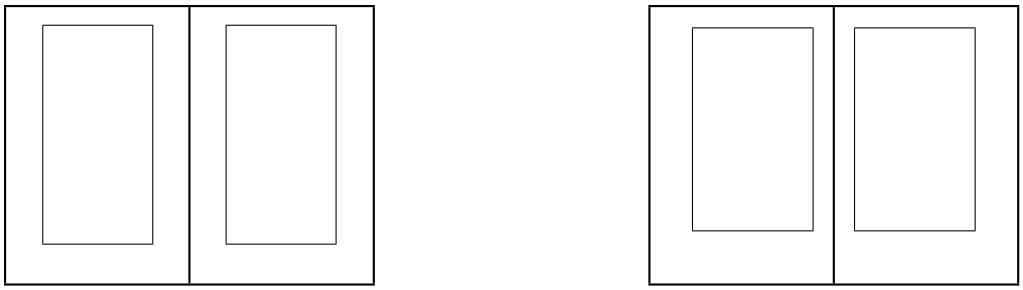


그림 2.9: 양면 펼침면: (왼쪽) 이태리, 1523. (오른쪽) 이태리 1499.

Brooke Crutchley의 *Two Men — Walter Lewis and Stanley Morrison at Cambridge*라는 책의 레이아웃을 보여준다. Crutchley는 캠브리지 대학의 출판 담당자였는데 매년 캠브리지나 타이포그라피, 또는 양자 모두에 대한 책을 만들어서 한정본으로 출판하였다. 이 책은 Monotype Barbon 폰트로 17.5 행간 26 행장의 253×162 사이즈이다.

오늘날의 학술 문헌 레이아웃을 그림 (2.8)에서 볼 수 있다. 이 책은 Douglas Schenck와 Peter Wilson 공저로 New York의 옥스포드 대학 출판부에서 1994년에 간행된 *Information Modeling the EXPRESS Way*이다. 이 책은 Computer Modern Roman 체 $10/12 \times 27$, 233×150mm 페이지로 만들어졌다. Ruari McLean의 *The Thames and Hudson Manual of Typography* (1988)이 그림 (2.8)의 오른쪽에 나타나 있다. 이 책의 조판은 Monophoto Garamond체 $10/11 \times 20$, 240×156mm로 이루어졌는데, 바깥쪽 여백을 넓게 잡아서 작은 삽화를 넣고 있다. 주석도 페이지 하단이 아니라 마진에 들어간다.

예전의 페이지 레이아웃은 컴퍼스와 자로 기하학적인 그림을 그려서 설계되었다. 사각형, 오각형, 육각형이 특히 많이 사용되었다. 특이하게도 그림 (2.9)의 왼쪽에 나타나 있는 편집영역은 페이지의 중앙에 놓여 있다. 편집영역은 정사각형을 이용해서 설계되었는데 세로길이는 가로길이(행장)의 두 배이다. 이 책은 Giangiorgio Trissino의 *Canzone*라는 책으로서 Ludovico degli Arrighi가 1523년에 로마에서 출판한 시집이다. 그가 디자인한 산문

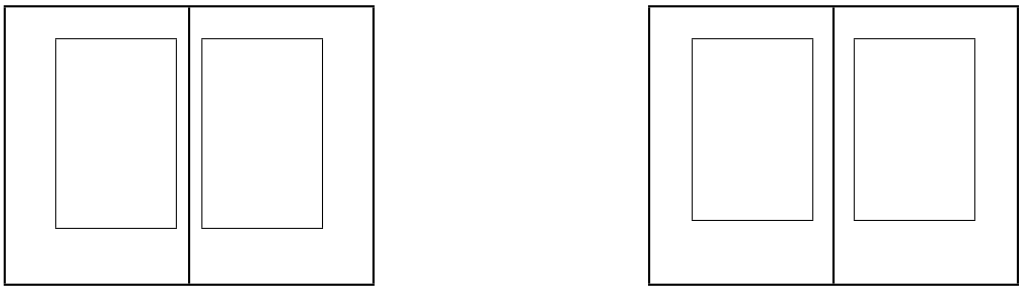


그림 2.10: 양면 펼침면: (왼쪽) 프랑스/포르투갈, 1530. (오른쪽) 구텐베르크, C15th.

서적은 바깥쪽 마진이 안쪽 마진보다 더 넓은 일반적 스타일을 따르고 있다. 그림 (2.9)의 오른쪽에 보인 페이지 종횡비는 단순한 3 : 2 비율이다. 편집영역의 종횡비는 1.7인데 이것은 오각형으로부터 유도된 것이다. 이 책은 Francesco Colonna의 *Hypnerotomachia Poliphili*라는 것으로 Venice의 Aldus Manutius 사에서 1499년에 출판하였다. 이 책에 대한 이야기가 Helen Barolini [Bar92]에 나오는데 원본의 일부를 싣고 있다.

1519년 포르투갈의 모험가 페르디난드 마젤란은 Sanlúcar de Barramada를 출발하여 스페인의 Cádiz까지 다섯 척의 배와 270명의 선원을 데리고 항해에 나섰다. 3년이 지난 후 한 척의 배와 18명이 귀환하였다. 이것이 최초의 세계일주가 되었다. 생존자 중에는 Antonio Pigafetta라는 사람이 있었는데 그는 이 모험에 대한 기록을 남겼다. 그의 기록 초고는 남아 있는 것이 극히 적다. 이 초고 가운데 하나의 레이아웃이 Beinecke Rare Book과 Manuscript Library at Yale에 나와 있는데 그것을 그림 (2.10)의 왼쪽에 보였다. 프랑스어로 쓰여진 초고는 *Navagation et descouurement de la Inde superieure et isles de Malueque ou naissent les cloux de Girosle* (정향나무가 자라는 상인도와 몰루카 섬들에 대한 항해와 발견)이라는 제목이 붙어 있고 아름다운 인문주의시대의 소문자로 쓰여졌다. 페이지당 27행이고 페이지 사이즈는 286×190mm의 양피지이다. 행장은 29.5, '행간'은 21 포인트이다. 바깥쪽 여백을 넉넉하게 두어 이야기의 중요한 부분을 나타내는 사이드노트로 활용하고 있다. 이 초고는 1530년 이전에 작성된 듯하고 서사(書士)와 작성장소는 알지 못한다.

요하네스 구텐베르크(1398–1468)와 그의 초기 계승자들이 만든 많은 책은 그림 (2.10)의 오른쪽에 보인 형태를 따르고 있다. 이 비율은 중세 고인쇄본(incunabula)[4]과 초고에 흔히 차용되기도 했다. 페이지와 편집영역의 종횡비는 모두 3 : 2로 동일하다. 마진의 비율을 2 : 3 : 4 : 6이다. 이 레이아웃을 기하학적으로 도안한 것을 그림 (2.18)에 보였다.

같은 책을 두 가지 버전으로 출판한 경우를 보여주는 것이 그림 (2.11)이다. 왼쪽 것은 1525에 쓰인 *Khamsch of Nizami*의 페르시아어 초고이다. 페이지 사이즈는 약 324 × 216mm이다. 삽화와 편집영역이 뒤엉켜 있다. 오른쪽 것은 이 초고 일부를 1975년에 뉴욕

[4]특히 1500년 이전에 인쇄된 책을 incunabula라고 한다.

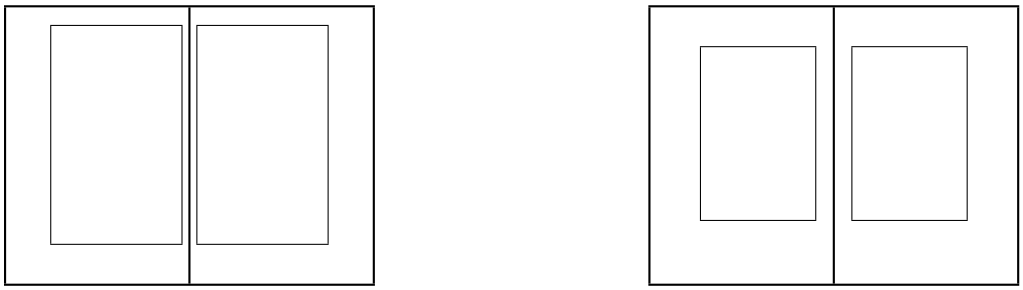


그림 2.11: 양면 펼침면: (왼쪽) 페르시아, 1525. (오른쪽) 미국 1975.

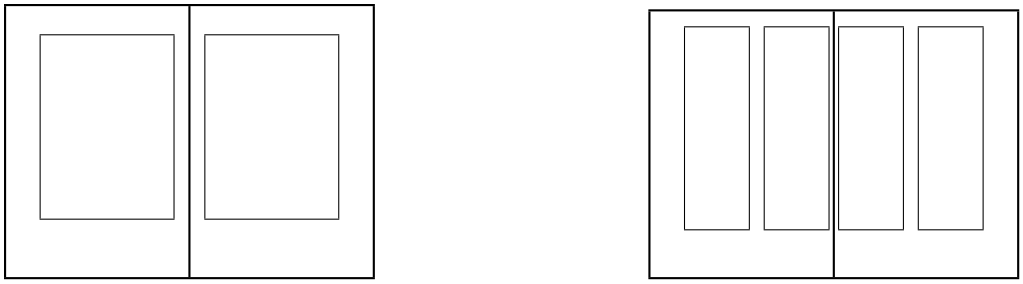


그림 2.12: 양면 펼침면: (왼쪽) 미국, 1952. (오른쪽) 영국, 1087.

의 메트로폴리탄 예술 박물관이 번역하여 출판한 *Tales from the Khamsch of Nizami*이다. 이 현대 버전은 300×200mm 페이지 사이즈로 원본보다 약간 작지만 종횡비는 동일하다. 편집영역은 32 파이카 행장에 15 포인트 행간으로 되어 있다.

Frederic Goudy는 저명한 미국인 글꼴 디자이너이다. 그림 (2.12)의 왼쪽 것은 1952년 캘리포니아 대학 출판부에서 낸 그의 책 *The Alphabet and Elements of Lettering*의 레이아웃이다. 이 책은 그가 디자인한 University of California Old Style 서체로 조판되었는데 여기에는 재미난 ct, st 리거쳐가 포함되어 있다. 행장은 36 파이카이고 행간은 18 포인트. 책의 전반부는 글쓰기와 폰트의 발전사에 대한 짧은 개관이고 나머지 반은 27장의 도판으로 이루어져 있는데 각각 알파벳 한 글자씩과 마지막에 앰퍼샌드 문자로 되어 있다. 이 도판은 각 글자들이 로마 시대에서 20세기 중반까지 어떻게 진화해왔는가를 보여준다.

그림 (2.12)의 오른쪽은 1087년에 영어로 쓰여진 수고(手稿) *Domesday Book*의 레이아웃이다. 1066년의 정복자 윌리엄 왕의 모든 영지를 기록하고 있다. 이 책은 Caroline 소문자, 2단 44행의 오른끝흘리기(raggedright)로 쓰여졌다. 두 개의 단은 폭이 약간 다르다. 책의 첫부분은 뒷부분보다 더 꼼꼼하게 기록되었는데, 아마도 필사자가 마지막을 서둘러 끝낸 것처럼 보인다.

그림 (2.13)는 ISO 국제 표준 $\sqrt{2}$ 종횡비에 해당하는 페이지 레이아웃 두 가지를 보여준다. 각각의 경우 편집영역은 동일하고 황금비를 따른다. 그러나 여백은 다르다. 왼쪽의

그림 2.13: 양면 펼침면: (왼쪽) ISO (1). (오른쪽) ISO (2).

그림 2.14: 양면 펼침면: (왼쪽) 영국, 1973. (오른쪽) LaTeX 10pt book 스타일.

그림 2.15: 양면 펼침면: (왼쪽) 미국, 1967. (오른쪽) 영국, 1982.

것은 바깥쪽 여백에 마진 노트를 위한 충분한 공간을 확보하고 있다.

또하나의 캠브리지 출판소의 크리스마스 북이 그림 (2.14)의 왼쪽에 나타나 있다. 이번에는 1973년에 출판된 Colin Franklin의 *Emery Walker — Some Light on his Theories of Printing and on his Relations with William Morris and Cobden-Sanderson*이다. 페이지 사이즈는 295×210mm, 행장은 31 파이카, 행간은 15 포인트이다. 오른쪽에는 LaTeX의 10pt book class로 US letterpaper에서 만들어지는 기본 레이아웃을 보였다.

1988년에 세상을 뜬 Adrian Wilson은 존경받는 미국의 북 디자이너였다. 북 디자인에 관한 그의 책 *The Design of Books*는 1988년에 출간되었고 Chronicle Books에 의해 1993년에 재간행되었다. 그림 (2.15)의 왼쪽이 이 책의 아웃라인이다. 2단 편집에 많은

그림 2.16: 양면 펼침면: (왼쪽) 영국, 1972. (오른쪽) 스위스, 1980.

그림 2.17: 양면 펼침면: (왼쪽) 영국, 1969. (오른쪽) 미국, 1989.

삽화가 들어 있고 letterpaper 사이즈로 된 책으로서 Palatino와 Linotype Aldus체, 행간은 25 포인트이다. 그리고 각 칼럼의 행장은 18 파이카이다. 다른 하나는 B. W. Robinson의 *Kuniyoshi: The Warrior Prints*라는 책으로 1982년에 Oxford의 Phaidon에서 간행된 것이다. 페이지 사이즈는 310×242mm이고 행장은 28.5 파이카, 13 포인트 행간으로 식자되었다. 안쪽 마진을 넓게 잡아서 일본 목판화 도면을 작게 싣고 있으며 그 가운데 어떤 것은 건너편 페이지까지 넘나들기도 한다. 이 책의 많은 부분이 전면을 차지하는 목판화 도면으로 되어 있는데 여기에는 캡션 외에는 텍스트가 나타나지 않는다.

Peter Eden이 쓰고 Warwick Hutton이 삽화를 그린 *The Waterways of the Fens*는 또 다른 캠브리지 크리스마스 북 가운데 하나이다. 17 포인트 행간에 27 파이카 행장이고 원본 페이지 사이즈는 195×150mm이다. 그림 (2.16)의 왼쪽에 나타낸 것이 이 책이다. 텍스트의 양이 페이지마다 다르고 많은 선화들이 들어 있다. 그림 가운데 일부는 펼침면 전체에 걸치기도 한다. 오른쪽 것은 또다른 화집으로서 Fedja Anselewsky가 쓰고 1980년에 Chartwell Books에서 출판한 *Dürer*라는 책이다. 이 책에는 텍스트보다 그림이 더 많은데 텍스트는 14 포인트 행간 23.5 파이카 행장의 2단 조판으로 되어 있다. 페이지 사이즈는 280×240mm로서 왼쪽 그림의 책에 비해서 월등히 크고 마진은 훨씬 좁다.

두 가지 화보집의 레이아웃을 더 보기로 하자. 그림 (2.17)에 있는 그림은 landscape 모드(즉 가로가 세로보다 긴 형태)로 그려져 있다. 그림의 형태가 페이지의 종횡비에 영향을 준 것이다. 왼쪽의 종횡비는 $\pi : e$이다. 행장은 통상보다 더 긴 37 파이카이고 행간은 17 포인트로서 역시 일반적인 경우보다 더 크다. 이 책은 Raymond Lister가 쓰고 Richard Bawden이 그림을 그린 *Hammer and Hand*라는 것으로 Centaur체로 조판되어 있다. 1969년 캠브리지 대학 출판부에서 간행된 또다른 캠브리지 출판부 크리스마스 북

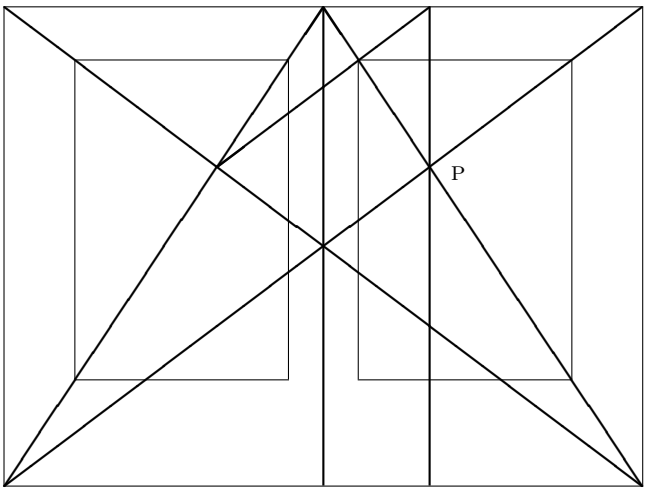


그림 2.18: 구텐베르크 페이지 디자인의 구성

이다. 그림 (2.17)의 오른쪽은 Peter Morse가 쓰고 1989년에 George Braziller가 출판한 *Hokusai — One Hundred Poets*이다. 도입부의 텍스트는 그림에 나타난 대로 2단조판되어 있다. 본문에 삽입된 일본 목판화의 원본은 대략 250×380mm 정도 크기의 오반(大版) 사이즈이다.

기하학적 구성

요즘 와서는 맘에 드는 페이지 종횡비를 선택하거나 계산하는 것이 어렵지 않다. 그러나 인쇄 시대 초창기에는 어떻게 했을까? 그들이 계산기를 사용하지 않았던 것은 확실하다. 그들에게 필요한 산수는 은행계좌를 유지하는 데 필요한 정도였을 뿐이다. 인쇄는 전문직이었고 장인들은 자신의 영업 비밀을 내놓고 드러내지 않았다. 내가 생각하기로 대부분의 디자인은 단순한 기하학 도형에 의하여 이루어진 것으로 믿는다. 즉 자와 컴퍼스 말고는 아무것도 사용하지 않았던 것이다.

Jan Tschichold [Tsc91, 44–57쪽]은 많은 구텐베르크의 책의 레이아웃을 단순한 도안으로 보여주었다. 그림 (2.18)은 이것을 보여주고 있다. 이 구성은 실제로 페이지를 9등분(그림에서 P로 표시된 점은 판면 전체의 대각선과 그 반인 페이지 대각선의 교점인데 페이지 전체와 편집영역 수직 길이의 1/3에 해당하는 점을 이루고 있다)한다. 이 구성은 페이지 종횡비가 어떻든 상관없이 상대적으로 일정한 결과를 내도록 되어 있다.

2.4 편집영역

편집영역은 단순히 텍스트로 이루어진 사각형 영역에 불과한 것은 아니다. 편집영역이

텍스트로 이루어져 있다면 이 영역은 다시 문단으로 나누어진다. 문단이 페이지보다 더 긴 것은 좋은 스타일이라 할 수 없다. 또, 편집영역이 표나 삽화를 포함하고 있다면 순수 텍스트와 대비되도록 해주어야 한다. 장이나 절의 제목줄을 포함하는 페이지라면 텍스트의 흐름을 적절하게 끊어주어야 한다. 일반적으로 편집영역은 텍스트, 공백, 텍스트 아닌 요소들이 혼합되어서 이루어진다.

삽화나 표가 없는 편집영역을 생각해보자. 텍스트의 행은 읽기 쉽도록 배열되어야 한다. 관행이나 최근의 심리학 연구에 따르면 행이 길면 일기가 어렵다고 한다. 따라서 편집영역의 폭에는 심리학적 상한이 있는 것이다. 사실 한 행이 너무 짧아서도 안되는데 그렇게 하면 텍스트를 양끝맞춤하기가 어려워지기 때문이다.

경험치에 의하면 단단(單段) 텍스트의 경우 행당 문자수가 60에서 70 정도인 것이 가독성이 높다고 알려져 있다. 이 범위는 45 내지 75 문자 정도로 보아도 될 것 같고, 66 문자가 이상적인 문자수로 간주되기도 한다. 이보다 짧으면 눈이 각 행을 따라 지나치게 빈번하게 움직여야 하고 이보다 길면 눈이 너무 멀리 건너뛰어야 하기 때문에 다음 행의 시작 위치를 찾아내기가 어려워져서 같은 행을 두 번 이상 읽거나 다음 행 한 줄을 읽지 않고 넘어가게 될지도 모른다. 2단 텍스트에서 이상적인 문자수는 45 문자를 전후로 5 글자 정도 범위이다.

Bringhurst [Bri92]는 주어진 폰트에서 행당 문자수를 결정하는 방법을 제시하고 있다. 알파벳 소문자의 길이를 잰 다음에 주어진 알파벳 길이와 행 길이, 행당 평균 문자수에 해당하는 카피피팅 조견표를 이용하는 것이다. 표 [2.2]는 Bringhurst의 카피피팅 조견표 축약본이다. 예를 들어보면 이 표는 폰트 소문자 길이가 130pt일 경우 행장은 단단에서 약 26파이카, 다단에서 18파이카가 적절하다고 제시하고 있다.

편집영역의 세로 높이는 모든 페이지에서 일정해야 한다. 맞쪽 두 페이지의 행은 가지런히 놓여야 한다. 즉 내지 한 장의 앞뒤의 같은 위치에 텍스트가 인쇄되어야 한다. 양면 맞쪽의 행이 가지런히 놓여야 한다는 것은 펼침면의 중앙을 기준으로 시선이 교착되지 않게 해야 한다는 것이고 내지의 앞뒤 같은 위치에 행이 찍혀야 한다는 것은 얇은 종이의 경우 뒷면이 배어 비칠 때 들쭉날쭉하지 않도록 해야 한다는 뜻이다. 그러므로 행간은 언제나 일정하여야 한다. 이것은 편집영역의 세로 길이가 각 행에 할당되는 공간의 정수배여야 한다는 것, 곧 행간의 배수로 정의되어야 한다는 것을 의미한다. 예를 들어 10포인트 글자면에서는 보통 2포인트를 행간에 더 주어서 12포인트를 행간으로 하는데 이것을 10/12와 같이 쓴다. 1파이카는 12포인트이므로 12포인트 행간은 파이카 단위로 손쉽게 표시할 수 있다(행당 1파이카). 또한, 삽화나 표 내지 장절 표제가 차지하는 공간은 행간의 정수배가 되어야 한다.

10포인트 활자를 행간없이 쓰는 경우를 10/10이라 표시할 수 있다. 이론적으로 활자의 자면은 d의 상단에서 p의 하단까지의 길이가 10포인트이다. 텍스트 행의 베이스라인에서 다음 행의 베이스라인까지의 거리도 10포인트이다. p라는 글자가 다음 행의 d 바로 위에 와서 겹칠 때를 생각해보자. 이 때는 b의 올림획(ascender)이 p의 내림획(descender)과 정

표 2.2: 행당 평균 문자수 조견표

Pts.	Line length in picas							
	10	14	18	22	26	30	35	40
80	*40*	**56**	**72**	88	104			
85	*38*	*53*	**68**	83	98	113		
90	*36*	*50*	**64**	79	86	107		
95	34	*48*	**62**	75	89	103		
100	33	*46*	**59**	**73**	86	99	116	
105	32	*44*	57	**70**	82	95	111	
110	30	*43*	55	**67**	79	92	107	
115	29	*41*	53	**64**	76	88	103	
120	28	*39*	*50*	**62**	73	84	98	112
125	27	38	*48*	**59**	**70**	81	94	108
130	26	36	*47*	57	**67**	78	91	104
135	25	35	*45*	55	**65**	75	88	100
140	24	34	*44*	53	**63**	73	85	97
145	23	33	*42*	51	**61**	**70**	82	94
150	23	32	*41*	*51*	**60**	**69**	81	92
155	22	31	*39*	*49*	58	**67**	79	90
160	22	30	39	*48*	56	**65**	76	87
165	21	30	38	*46*	55	**63**	74	84
170	21	29	37	*45*	53	**62**	72	82
175	20	28	36	*44*	52	**60**	**70**	80
180	20	27	35	*43*	51	59	**68**	78
185	19	27	34	*42*	*49*	57	**67**	76
190	19	26	33	41	*48*	56	**65**	74
195	18	25	32	40	*47*	54	**63**	72
200	18	25	32	39	*46*	53	**62**	**70**
220	16	22	29	35	*41*	*48*	56	**64**
240	15	20	26	32	38	*44*	51	58
260	14	19	24	30	35	41	*48*	54
280	13	18	23	28	33	38	*44*	50
300	12	17	21	26	31	35	41	*47*
320	11	16	20	25	29	34	39	*45*
340	10	15	19	23	27	32	37	42

확하게 만나게 될 것이다. 이것을 피하기 위해서 베이스라인 사이의 간격을 활자 크기보다 조금 더 주는 것이다. 2포인트의 여분을 주게 되면 이 두 행이 조금 떨어지게 되므로 행간을 12포인트로 한다. 추가행간 없이도 잘 읽어지는 폰트는 거의 없다. 전형적인 설정은 9/11, 10/12, 11/13, 12/15로 하는 것이다. 행장이 길어지면 행간도 조금 늘어나야 한다. 그리고 가늘고 작은 폰트보다 두껍고 큰 폰트일수록 더 많은 여분이 필요하다. 텍스트에 상첨자나 하첨자가 많이 스이거나 대문자가 많은 경우에도 행간을 조금 늘려주는 것이 좋다.

페이지 색조

타이포그라퍼가 해야 하는 일 중의 하나는 '색조' 면에서 모든 페이지들이 일관성있게 되도록 하는 것이다. 이것은 편집영역이 얼마간 균일한 회색조를 띠도록 한다는 의미이다. 너무 많은 공백(white space)을 남기면 독자들의 주의를 산만하게 할 수 있다. 표제 주변에는 공백이 와도 괜찮을 것이다. 표제 자체가 주의를 끌려는 것이기 때문이다. 문단 사이에도 공백을 넣을 수 있을 것이다. 그러나 이것은 디자이너가 선택할 문제이다. 단어 간격으로 들어간 공백 위치가 이웃하는 행끼리 만나게 되면 세로로 길게 공백이 이어진 이른바 개천(rivulet) 현상이 일어난다.[5] 이것을 해결하려면 수작업이 필요하다. 저자는 공백의 위치를 바꿀 수 있도록 단어 선택을 수정해거나 해야 하고 조판사는 공목의 위치를 조금씩 조정해 보아야 한다.

또다른 나쁜 예로 너무 많은 행이 하이픈으로 끝나거나 여러 개의 인접하는 행이 같은 단어로 시작하거나 끝나는 경우를 들 수 있다. 이것은 개천을 만들뿐 아니라 다음 행과 혼동을 일으켜 텍스트를 제대로 읽을 수 없게 만든다.

사용된 중심 폰트는 페이지의 색조를 결정한다. 특정한 폰트를 사용했을 때 어떤 색조를 띠는지 확인하려면 적당히 긴 보통 텍스트로 한 페이지 정도를 만들어 보아야 한다. 폰트 패밀리가 달라지면 색조도 달라지고 같은 가족의 여러 폰트를 섞어 써도 역시 그렇다. 예컨대 Computer Modern Roman, Italic, Sans 폰트들을 같은 페이지에 섞어써서 색조를 직접 확인해보라. Rogers [Rog43], Lawson [Law90], Dowding [Dow98], Morison [Mor99]의 책들은 많은 다양한 폰트를 섞어써서 조판한 페이지를 보여준다.

가독성

타이포그라피의 주된 요구사항은 그 문서가 가독성이 있어야 한다는 것이다. 가독성이란 일정한 조건 하에서 쉽게 읽어지는 정도를 가리키는 말이다. 예컨대 버스 옆면에 부착한 광고 포스터의 가독성 기준을 안락의자에 앉아 책을 읽는 경우에 적용할 수 없는 것이다. 근본적으로 독자는 문서를 읽을 때 책 모양 때문에 육체적 긴장을 느껴서는 안된다. 그렇

[5] 판면이 세로로 갈라지는 것을 말하는 것으로 보통 lizard라고 하는데 여기서는 rivulet이라 하였다—역주.

지만 물론 내용으로부터는 날카로운 정신적 긴장에서 극단적 지겨움까지 어떤 것이라도 느낄 수 있다.

글꼴의 모양과 편집영역의 레이아웃은 가독성과 '미적' 표현 양자를 최적결합하는 것이어야 한다. 문서의 디자인 자체는 거의 두드러지지 않으면서 저자가 전달하고자 하는 것을 전적으로 뒷받침해주는 것이어야 한다. 그렇지만 만약 당신이 Hermann Zapf [Zap00]와 같은 대가라면야 이런 규칙에 얽매이지 않아도 좋다.

활자의 자면

현재까지 쓰이는 유럽 문자의 최초 형태는 석비에 새겨넣은 그리스의 새김글씨(inscription)였다. 날카로운 것으로 쪼아서 자유롭게 새겨넣었다. 시간이 지나자 글씨는 보다 두꺼워지고 세리프가 나타나기 시작했다. 로마인들은 이 후대의 문자 형태를 취하였다. 처음에 돌에 글자를 새길 때 그들은 넓고 평평한 붓을 사용했다. 이것은 자연히 붓을 움직이는 힘과 방향에 따라 획의 각도가 달라져서 세리프가 생기게 하고 글자 획의 두께가 달라지게 만들었다.

로마 시대에서 구텐베르크까지 유럽에서는 문자 형태에 많은 변화가 있었다. 필사자들은 제목, 부표제, 연속 텍스트, 첫글자 장식 등에 서로 다른 서체를 사용하였다. 그 후 서체에는 두 종류가 발전하였는데 그것을 각각 대자*(majuscules)*, 소자*(miniscules)*라고 한다. 대자는 더 크고 더 공식적인 것으로 쓰였고 소자는 좀 작고 형식에 덜 구애되었다. 요즘 우리들은 이 두 종류를 각각 대문자, 소문자라고 부른다. 대문자는 로마 시대로부터 유래하는 것인 반면 소문자는 약 천년 후인 신성로마제국 샤를마뉴 치세에 그 기본적 골격이 갖추어졌다. 더 세분된 구분도 나왔는데 흑자(보통 고딕체 또는 옛 영어체라고 하는 것)와 로만 활자가 그것이다.

이 활자들은 모두 곧추선 모양이다. 이탤릭 서체는 16세기 초반에 이탈리아에서 보다 장식적인 글자체로 등장하였다. 처음에는 소문자뿐이었지만 대문자 로만체와 함께 쓰이기 시작했다. 16세기 말경에 이르러 기울어진 로만체 대문자도 이탤릭체와 함께 사용되고 있다.

19세기 후반에 산세리프 활자형태가 나타났다.

세리프 체와 산세리프 체를 주의깊게 관찰하면 세리프에는 세 가지 주요 기능이 있음을 알 수 있다.

1. 각 글자들이 엉키지 않도록 한다.

2. 그러면서도 글자들이 하나의 단어에 모여 있게 해준다. 그 결과 가독성을 높이는 데 기여한다. 연구결과에 의하면 우리가 글을 읽을 때는 글자 단위로 읽는 것이 아니라 단어의 형태를 한꺼번에 인지하는 경향이 있기 때문이다.

3. 비슷하게 생긴 글자들을 구분할 수 있게 해준다.

오랜 경험을 통해 볼 때 세리프 폰트가 산세리프 폰트보다 더 읽기 쉬운 것으로 보인다.[6] 특히 텍스트가 불명료할 때 더 그런 경향이 있다. 스스로 실험해보라. 어떤 구절을 한번은 산세리프로 쓰고 또한번은 세리프 폰트로 쓴 다음에 윗 부분 반 정도를 가리고 그 구절이 무엇인지를 알아내어보자. 이번에는 그 구절의 아래쪽 반을 가리고 역시 같은 일을 되풀이해본다. 어느 쪽이 읽기 쉬운가? 다음에 몇 글자를 보이겠다. 먼저 산세리프이다.

a c l m n p q o

로만 글꼴이다.

a c l m n p q o

산세리프 폰트는 가끔 그 뜻을 알려면 전후 맥락의 파악이 필요할 때가 있다. 예를 들어 [McL80] 다음과 같은 것이 있다면,

lll

이것이 'lll'인지 백십일인지 '삼'인지 아니면 완전히 별개의 딩뱃 기호인지 크리킷 기둥인지 구별할 수 있겠는가.

이어지는 텍스트를 조판하는 경우 가독성을 높이는 데 일반적으로 받아들여지는 세 가지 원칙이 있다.

1. 산세리프 글꼴은 내재적으로 세리프 글꼴보다 가독성이 낮다 [Whe95].

   여기에 대해서는 이미 언급했다. 세리프 글자체는 산세리프 글자체보다 더 다양하다. 그리고 세리프가 있으면 글자들을 단어 안에 묶어주는 것과 같은 다른 기능도 수행한다.

   이것이 산세리프 글자꼴이 언제나 로만체보다 가독성이 낮다는 의미는 아니다. 잘못된 세리프 글꼴이 적절하게 사용된 산세리프보다 훨씬 가독성이 낮다. 염두에 두어야 할 것은 일반적으로 말해서 산세리프와 관련되어 가독성을 낮추는 요소가 있다는 정도일 뿐이다. 보편적인 연속되는 문장을 읽을 때 좋은 세리프 폰트를 쓰는 것이 좋은 산세리프보다 눈에 더 쉽게 들어오는 경향이 있다는 말이다.

2. 잘 디자인된 로만 대소문자가 그 변형보다 더 읽기 쉽다.

   이것은 예외가 많은 원칙이다. 변형이라 함은 이탤릭이나 볼드체를 말한다고 생각할 수 있다. 이런 서체는 예를 들면 일부 텍스트의 강조 강조와 같은 특별한 목적을

[6] 이것은 사실 논란의 소지가 있는 주장으로 충분히 익숙해지면 산세리프가 더 읽기 쉽다고 주장하는 사람도 있다.

위해 디자인된 것이지 본문의 가독성을 높이기 위한 것이 아니다. 그렇지만 어떤 이탤릭 글꼴은 그 대응 로만 서체보다 가독성에서 떨어질 것이 없는 경우도 있다. 17세기에는 많은 책들이 이탤릭만의 본문으로 만들어지기도 했지만 요즘 우리는 로만 서체에 익숙해져 있다.

3. 행간격보다 단어간 간격이 더 가까워야 한다.

   텍스트는 잉크와 공백이 섞여서 짜여지는 것이다. 독시할 때 눈은 공백을 건너뛰는 경향이 있다. 두 개의 공백이 나오면 그 중에 작은 것을 무시하고 건너뛰려 한다. 단어 간격이 행간격보다 크면 한 줄이 끝나기도 전에 다음 줄로 건너뛰는 자신을 발견할 수 있을 것이다.

   또한, 행이 너무 길면 한 행이 끝난 위치에서 다음 행 시작 위치까지 눈이 되돌아오는 데 어려움을 겪게 될 것이므로 다음 행의 시작 위치를 잘 잡지 못하게 될 수 있다.

   텍스트 행은 단어 간 간격을 조금씩 변경하여 양끝맞춤을 한다. 그래도 간격이 잘 안 맞으면 행의 마지막 단어를 하이픈처리하는 경우도 있다. 가끔 산세리프 글꼴은 오른끝 흘림(ragged right)으로 배열했을 때 보기 좋은 경우가 있는데 이 때는 단어간 간격이 일정하게 유지된다. 단(段)이 좁은 경우에도 오른끝흘림으로 조판하는 것이 좋을 수 있다.

세리프 폰트와 산세리프 폰트

앞서 언급했듯이 세리프와 산세리프 글꼴에 대한 문제는 항상 논쟁거리이다. 특정한 작업에 어떤 것이 가장 좋은가에 대한 자신의 견해를 가지게 될 것이다. 그러나 여기서는 이 주제에 대한 문헌 몇 가지를 살펴보는 정도의 일반적 언급만을 하겠다.

Bohle [Boh90] 에 따르면, 독자들이 본문 서체로 로만체를 선호하는 것은 그들이 그 글꼴에 익숙해져 있기 때문이다 [Reh72]. 로만 서체는 세리프가 글자들을 이어주어서 단어 형상으로 연결해주기 때문에 산세리프보다 더 읽기 쉬울 수 있다 [Reh72].

Craig [Cra92] 는 활자의 세리프가 수평 흐름을 촉진하여 편안한 독서를 가능하게 한다고 하였다.

Degani [Deg92] 의 한 연구에서 비행조종사들이 체크리스트를 위기상황에서 읽는 경우 산세리프 서체가 세리프 서체보다 더 나았다고 한다.

Schriver [Sch97] 에 따르면, 세리프와 산세리프가 독자에게 선호되는 정도에 차이는 없다 [HR83, Tin63]. 그리고 읽는 속도도 비슷하다 [Gou87, HR83, Zac69]. 세리프 활자는 길게 이어진 문장에서 산세리프보다 더 읽기 쉬울 수 있다 [Bur59, HK75, Whe95].

Wheildon [Whe95] 는 오스트레일리아 시드니에 살고 있는 독자 250명을 상대로 한 일련의 연구에서 그들에게 세리프와 산세리프 중 어느 쪽이 나은지 물었다. 그의 결론 가운데 일부를 인용하면,

- 세리프 본문이 사용된 경우가 산세리프 본문의 경우보다 이해도에서 다섯 배나 높았다.
- 대문자 글자들의 위쪽 반이 아래쪽 반보다 더 인지도가 높았다.
- 헤들라인(절제목 등)의 경우 세리프체와 산세리프체로 식자했을 때 또는 로만체와 이탤릭체로 식자했을 때 가독성의 차이는 거의 없었다.
- 대문자로 식자된 헤들라인은 소문자로 식자된 경우에 비해 유의할 정도로 가독성이 낮았다.

연구조사의 결과는 장문의 연속하는 본문에는 세리프체가 유리하다는 쪽으로 기울어져 있는 것 같다. 그러나 표제행에 대해서는 어느 것을 선택하든 큰 차이 없는 것 같다.

고아와 과부

독자가 책을 죽 읽어내려가다가 신경쓰이는 페이지 나눔을 만나면 독서에 방해를 받게 된다. 이런 페이지 나눔은 주로 문단의 처음이나 끝이 페이지 나눔 근처에 걸려 있을 때 발생한다.

과부라는 것은 한 단락의 마지막 행이 그 페이지의 첫째 줄로 되어 있는 경우이다. 이 용어는 한 문장의 마지막 단어 하나만으로 한 줄이 되고 끝나는 경우에도 이따금 쓰인다. 과부는 인생에 희망이 없다. Robert Bringhurst의 말을 빌자면, '과부란 과거만 있고 미래가 없다.' 조판에 있어서 과부는 피해야 한다. 특히 한 챕터가 끝난다든지 하여 그 페이지에 단 하나의 행밖에 없을 때는 반드시 피해야 한다. 한 장이 끝나는 페이지에서 합당한 최소 행수는 적어도 다섯 행은 넘어야 한다는 것이다.

고아는 타이포그라퍼에게 과부만큼 성가신 것은 아닌 듯하다. 고아란 한 단락의 처음 한두 행이 페이지의 바닥에 남는 경우를 말한다. Bringhurst의 기억을 위한 트릭에 따르자면, '고아는 미래는 있지만 과거가 없는 것'이다.

문단과 문단 첫머리 장식

초창기에는 오늘날 우리가 보고 있는 것과 같은 문단이라는 것이 없었다. 텍스트는 죽 이어서 쓰여졌고 중요한 구분이 이루어져야 할 때, 예컨대 성경의 새로운 책이 시작되는 것과 같은 경우에만 단락을 끊었다. 그 대신 필사자들은 문단 시작을 표시하기 위해 ¶(필크로, pilcrow)와 같은 기호를 사용했다. 이 기호는 그리스어 *parágraphos*의 첫 글자

Π에서 나온 것이다. 문장부호도 전혀 사용되지 않았고 대문자도 사용하지 않았다는 점을 생각하라. 말하자면 다음과 같은 모양을 보면서 짐작해볼 수 있을 것이다.[7]

> usque ¶ te canit adcelebratque polus rex gazifier hymnis ¶ transzephyrique globum scandunt tua facta per axem

요즘은 문단 끝을 나타내기 위해 행을 중단하고 새로운 행으로 다음 문단을 시작하여 구별한다. 문제가 있다. 만약 한 문단의 마지막 행이 그 행장을 모두 채우는 경우에는 다음 문단이 새로운 문단의 시작임을 어떻게 구별하는가? 두 가지 해결책이 있는데 불행히도 이 두 가지 방법이 다 쓰이는 경우도 볼 수 있을 것이다. 각 문단의 첫번째 행을 얼마간 안으로 들여쓰는 방법이 그 하나이고 다른 하나는 앞 문단의 마지막 줄과 그 다음 문단 사이에 수직 간격을 조금 더 벌리는 방법이다.

수세기 동안 사용되어 온 전통적 테크닉은 문단 첫 행 들여쓰기 방법이다. 들여쓰기는 굳이 깊지 않아도 된다. 1em 정도면 충분하다. 그러나 편집영역의 가로 길이가 길다면 조금 더 들여쓰는 것이 좋을 것이다.

문단 간격을 넓히는 다른 방법은 주로 상업 우편물에 쓰이는 것으로 최근의 발명품이다. 문단의 첫 줄은 들여밀기하지 않고 그 대신 문단 사이에 빈 줄을 하나 둔다. 이것은 타자기를 사용할 때 허용할 수 있는 방법이다. 그러나 실제의 맞춰짜기가 아름답게 이루어진다는 보장이 없는 데다가 한 문단이 끝나는 곳이 행장을 다 채우면서 동시에 그 페이지의 끝인 경우에는 문제가 된다. 그 다음 문단은 다음 페이지 꼭대기에서 시작하는데 두 문단을 구분하는 빈 행이 사라져버리는 것이다. 독자는 페이지 나눔 전후에 새로운 문단이 시작되었는지 어떤지 알 수 없게 된다.

장절 표제 이후의 첫번째 문단은 그것이 새로운 문단임을 지시할 필요가 없다. 문단의 위치 자체가 자명하기 때문이다. 그래서 장절 표제 이후의 첫번째 문단은 들여쓰기하지 않는다. 일부 소설에서는 장 표제만이 표시되고 각 장을 절로 나눌 때는 절 사이에 빈 행만을 두는 경우가 있다. 들여쓰기 하지 않는 문단의 경우와 같이 이 때도 새로운 절이 페이지 나눔과 만나는 경우 문제가 된다. 그래서 타이포그라퍼들은 이따금 절 구분을 위한 장식을 사용한다(예를 들면 별표를 몇 개 가운데 나열하는 것 등).

Some typographers like to start the first paragraph in a chapter with a versal. 타이포그라퍼 중에는 챕터 안의 첫번째 문단을 문단머리 장식으로 시작하는 것을 좋아하는 사람도 있다. 문단머리 장식*(versal)*은 첫 글자를 크게 위로 올려 쓰거나 아래로 내려 쓰는 것을 말한다. 이것은 초고의 첫 글자를 강조하기 위해 필사자들이 쓰던 전통으로부터 나온 것이다. 문단머리 장식은 위로 올려 쓸 수도 있고 아래로 내려 쓸 수도 있다. 때로 마진에 둘 수도 있다. 또 다른 방법으로 처리하는 것도 가능하다.

[7] 꼭 그렇지는 않을 수도 있다. 이 두 '문단'은 라틴어의 ABC 초급용 문장이다.

Some versals, especially dropped versals, are very difficult to typeset correctly. 문단 머리 장식은 특히 그것이 아래로 내려진 것일 때 정확하게 식자하기가 매우 어렵다. 이것을 제대로 하기 위한 여러 가지 시도들이 대부분 실패를 맛보았다는 사실을 경고해 둔다. 예를 들어 이 두 문단 첫머리의 끌어내려진 문단머리 장식을 비교해보라. 이 두 글자의 폰트는 동일하지만 앞서 나온 것은 이 문단 첫머리에 있는 것과 비교해보았을 때 흉하다.

It is easier to start a paragraph with a raised capital than one that is dropped. 새로운 문단을 시작하는 대문자를 끌어내리지 않는 것이 더 쉽다. 끌어올려진 문단머리 장식은 문단 사이에 자연스러운 약간의 공백이 있어야 사용할 수 있다. 보는 바와 같이 추가 공백이 이 문단 앞에 삽입되어야 문단머리 장식이 잘 들어간다. 끌어올린 문단머리 장식에도 조판상의 문제가 있지만 이것은 끌어내림 문단머리 장식보다 더 미묘한 것이라서 독자는 거의 문제를 알아차리지 못한다.

【memhangul】 위의 세 문단은 첫번째 문장을 그대로 보였다. 한글 문서에서 문단머리 장식이 과연 조판상 어떻게 구현되어야 하고 어떤 효과가 있을 것인가에 대해서 역자는 아직 확신하고 있지 못하고 이것은 전적으로 영문 문서의 조판 관행이라고 판단하기 때문이다. 그러나 굳이 한글 문서에서 이러한 문단머리 장식을 흉내내면 어떻게 되는가를 다음 문단에서 보였다. ■

작은 대문자가 문단머리 장식 뒤의 몇 단어에 쓰이는 것이 일반적이다. 이것은 큰 문단머리 문자에서 그 뒤의 보통 본문으로 옮겨가는 과정에 해당하는 것이다. 한 행 전체가 작은 대문자로 식자되어서는 안된다. 그러면 문단의 나머지 부분과 두드러지게 구분되어 버리기 때문이다.

Another way of starting a paragraph is to use small caps for the first few words. 문단을 시작하는 또다른 방법은 처음 몇 단어에 작은 대문자를 쓰는 것이다. 폰트의 차별은 새로운 문단의 시작을 잘 보여주지만 문단머리 장식에 비하여 훨씬 온건하다. 작은 대문자가 아니라 보통 크기 대문자를 쓰게 되면 소문자와의 대조가 너무 두드러진다.

각주

각주는 편집영역의 일부로 간주된다. 편집영역을 벗어나 아래쪽에 놓이는 하단 면주와는 달리 각주는 편집영역의 일부 공간을 할당받아서 식자된다.

각주는 보통 편집영역에 쓰인 것과 같은 글꼴을 사용한다. 즉 업라이트 세리프 폰트를 본문에서 사용하였다면 각주에도 같은 글꼴을 쓴다. 글꼴 크기는 조금 작게 하여 주석을 본문과 구별하고 행간도 조금 줄여준다. 각주 영역의 맨 마지막 행은 편집영역의 마지막 행과 같은 위치에 식자되어야 한다. 이를 위해서 각주 앞의 수직 공백을 약간씩 조절할 필요가 있다.

각주를 본문과 분리하기 위해서 빈 줄을 넣는 경우가 있고 때로는 구분을 위해 길지 않은 괘선을 긋기도 한다.

2.5 폴리오

*folio*라는 단어는 동음이의어이다. 이것은 책 속의 내지 한 장(앞뒤 두 페이지로 이루어지는)을 뜻하기도 하고 책의 크기를 가리키기도 하고 접지로 이루어진 책(셰익스피어의 첫번째 중철본이라고 할 때와 같이)을 가리키기도 하고 책의 페이지마다 찍어주는 면번호를 가리키기도 한다. 여기서는 마지막 의미로 사용하겠다.

책에는 면번호를 두어야 한다. 최소한 독자가 현재 어디를 읽고 있는지 알려주는 것이다. 가끔 면번호를 책매김쪽(안쪽)에 두는 경우가 있는데 이 위치는 책을 훑어보는 데 도움이 되지 않는다. 보다 일반적인 위치는 편집영역 기준으로 가운데이거나 편집영역의 바깥쪽에 두는 것이다. 심지어 여백 밖에 두는 경우도 있다. 면번호는 페이지의 상단이나 하단 어느 쪽이든 상관없지만 적어도 장 표제가 오고 장이 시작하는 페이지에서는 페이지 하단의 중앙에 두는 것이 일반적이다. 그래야 표제 텍스트와 혼동을 일으키지 않는다.

책의 모든 페이지는, 표제지와 약표제지를 제외하고 모두 번호를 매긴다. 면번호를 표시하지 않더라도 번호매김은 이어진다. 책의 앞부분 면번호는 로마 숫자로 표기하는 것이 보통이고 본문과 뒷부분은 아라비아 숫자로 앞부분이 끝난 다음을 1로 해서 일련번호를 붙인다. 어떤 학술 문서의 경우 면번호가 장번호와 페이지 번호를 모두 표시하는 형태인 경우가 있는데 이 때는 새로운 장이 시작하면 페이지 번호도 1에서부터 새로 시작한다. 다른 형식의 면번호 붙이기도 가능하겠지만 보편적인 것은 아니다.

면번호는 편집영역과 페이지 마진을 고려하여 보기좋게 위치해야 한다. 면번호에 사용되는 폰트가 편집영역의 폰트와 같을 필요는 없다. 그러나 최소한 보조적인 것이어야지 너무 튀는 것이어서는 안된다.

2.6 헤더와 푸터

헤더와 푸터는 각 페이지의 상단과 하단에 반복해서 위치하는 요소이다. 대표적인 것은 면번호로서 이것도 헤더나 푸터이지만 (항상 그런 것은 아니라 해도) 편집영역의 첫째 줄 또는 그 아래 위치하는 경우도 있다.

이제부터 헤더와 푸터를 굳이 구별하지 않고 헤더라고 지칭하겠다. 헤더는 이따금 순수하게 장식적으로만 쓰이는 경우가 있다(면번호는 제외). 수평선을 긋는다든가 그밖의 텍스트 외적인 약물을 사용한다든가. 일반적으로는 독자가 현재 읽고 있는 부분이 어디인지를 알려주는 기능을 가진다.

가장 견고한 헤더는 문서 자체의 제목을 적는 것이다. 이것은 다른 내용으로 바뀌지 않는 것으로서 기능적이라기보다 장식적 의미가 강하다. 독자가 자신이 읽고 있는 문서

의 제목을 매 페이지를 넘길 때마다 상기해야 할 필요가 있겠는가. 문서의 현재 위치, 예컨대 장 표제나 장 번호를 알려주는 헤더는 좀더 유용하다. 책을 내려놓았다가 나중에 집어들고 계속 읽으려 할 때 이런 헤더들은 이전에 읽던 곳을 찾는 데 도움을 준다. 어떤 이유에서든 이전 장을 찾아보려 할 때도 장 표제가 펼침면에 나와 있으면 찾기가 쉽다. 최소 기능적 헤더는 문서 표제를 한쪽 편에 싣고 다른 편에는 장표제를 싣는 것이다. 좀더 학술적인 문헌에서는 장표제와 절표제를 각각 헤더에 넣어두는데 이것이 보다 유용하다.

이따금 헤더와 푸터가 모두 사용되기도 한다. 이럴 경우 어느 하나는 고정된 텍스트를 표시하는데 일반적으로 저작권 고지를 넣는 경우가 많다. 내 생각에 이것은 문서 일부가 복사되거나 복제되는 것을 싫어하는 출판업자에게만 유용하다고 생각된다.

헤더는 편집영역의 책등쪽 기준선을 기준으로 정렬되는데 편집영역의 상단 중앙에 오기도 한다. 어떤 경우든 면번호와 뒤엉켜서는 곤란하다. 활자체가 본문 서체와 동일할 필요는 없다. 예를 들어 헤더에 이탤릭이나 작은 대문자를 사용할 수도 있는데 다만 본문의 조판 스타일과 어울리도록 하면 된다.

제 3 장

까다로운 문제들

3.1 들어가는 말

좋은 타이포그라피를 판단하는 주요 기준은 가독성과 페이지 색조이다. 이 장에서는 이 주제에 관련된 소소한 문제 일부를 검토해보겠다.

3.2 단어간격과 행간격

연구결과에 따르면 숙련된 독자는 책을 읽을 때 글자를 하나하나 읽는 것이 아니고 단어 전체의 윤곽을 한꺼번에 인식한다고 한다. 단어 사이에 아주 좁은 간격만 주어도 단어를 구분하는 데 충분하다.

대부분의 타이포그라퍼들은 길게 이어지는 문장에서 단어 사이에 놓이는 공백의 크기는 글자 'i'의 폭 정도여야 한다고 말한다. 이보다 더 짧으면 단어를 한꺼번에 보려 하게 되고 이보다 너무 길면 페이지 색조에 흰 점이 너무 많아지는 데다가 눈의 움직임이 행을 따라잡기보다 다음 행으로 넘어가려는 경향이 강해진다. 그림 (3.1)은 단어간격 값을 변화시킨 예를 보여준다.

흰 점이 발생하는 것을 피하기 위해서 타이포그라퍼들이 권장하는 것은 구두점 뒤에 여분의 공백을 두지 않는 것이다. 이것이 각 나라마다 발전해온 타이포그라피 관행이나 개인적 취향에 좌우되는 것이기는 하지만. 나는 타자 원고에서 문장 종지 뒤에 두 개의 스페이스를 넣는 것이 언제나 눈에 거슬렸다. 그렇지만 조판이 완료된 텍스트에서 추가 공백은 대개 그보다는 작았다.

행과 행 사이의 간격은 단어간 간격보다는 커야 한다. 그렇지 않으면 시선이 다음 단어보다 다음 행으로 건너뛰는 경향이 생긴다. 그림 (3.2)는 서로 다른 행간격으로 조판된 결과를 보여준다. 단어 간격은 일반적인 크기로 하였다.

The following paragraph is typeset with double the normal interword spacing for this font.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

The following paragraph is typeset with the normal interword spacing for this font.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

The interword spacing in the following paragraph is the width of the letter ‘i’.

Most typographers state that the space between words in continuous text should be about the width of the letter ‘i’. Any closer and the words run together and too far apart the page looks speckled with white spots and the eye finds it difficult to move along the line rather than jumping to the next word in the next line. Extra spacing after punctuation is not necessary.

그림 3.1: 단어 간격

This paragraph is set solid — the interline spacing is the same as the font size. The normal interword spacing is used. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

This paragraph is set with the normal interline spacing for the font. The normal interword spacing is used. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

This paragraph is set with the interline spacing 20% greater than is normal for the font. The normal interword spacing is used. The spacing between lines of text should be greater than the interword spacing, otherwise there is a tendency for the eye to skip to the next line instead of the next word.

그림 3.2: 행간격

【memhangul】 한글 문서는 영문 문서의 경우와는 다른 사정이 있다. 한글 한 글자는 어쨌든 영문자 두 자 안팎의 크기에 해당하고 베이스라인이 영문자의 경우보다 아래 있어서 자면을 가득 채우는 경우가 많으므로 만약 영문과 동일한 행간 및 어간을 사용한다면 페이지 색조 면에서 훨씬 어두워진다. 따라서 행간과 어간을 영문보다 넓혀주어야 할 것이다. 어간의 경우는 반자(.5em)보다는 적고 영문의 기본 어간보다는 넓은 적절한 값을 사용하는 것이 좋다고 생각하고 있다. memhangul-ucs 패키지는 기본 행간을 `stretch=1.333`으로 잡고 있다. 그러나 이 값은 조판 사정에 따라 적절하게 달라져야 할 것이라고 생각한다. ■

3.3 약어, 약성어

영어의 약어 쓰는 방식은 마침표(period)를 약어 뒤에 찍는 것이다. 예외는 약어의 마지막 글자가 원 단어의 마지막 글자와 같을 경우이다. 따라서 미스터는 Mr로 닥터는 Dr로, 그리고 프로세서는 Prof.로 표기한다. 이 경우의 마침표 뒤에는 여분 공백을 두면 안된다. 여분 공백은 문장의 마침의 경우에 쓰인다.

약성어(acronym)는 대문자로 조판한다. 문제는 어떤 대문자를 쓸 것이냐이다. 간단한 방법은 UNICEF와 같이 보통 글꼴의 대문자를 그대로 쓰는 것이다. 그러나 만약 너무 많은 약성어가 흩어져 있으면 반점 현상이 나타나서 얼룩덜룩해질 수 있다. 폰트 패밀리가 작은 대문자를 지원한다면 작은 대문자를 사용하는 방법이 있다. 예컨대 UNICEF. 작은 대문자를 사용할 수 없거나 바람직하지 않다면 보통 대문자를 작게 해서 쓰는 방법이 있

다. 예를 들면 UNICEF나 UNICEF와 같이 하는 것이다. 적절한 사이즈를 선택하려면 많이 시험해보아야 한다.

3.4 줄표와 줄임표

대부분의 폰트는 최소한 세 가지 길이의 줄표(dash)를 갖추고 있다. 가장 짧은 것이 하이픈(-)이다. 그리고 'n' 자 길이에 해당하는 짧은 줄표 en-대시(–)가 있고, en-대시의 두 배 길이 정도인 em-대시(—)가 가장 길다. 좀더 전문적인 폰트는 더 다양한 줄표를 제공한다.

당연한 일이지만 하이픈은 하이픈처리에 쓰인다. 예컨대 em-dash와 같은 곳이나 단어를 문장 끝에서 자를 때 사용된다.

en-대시는 숫자의 범위를 나타내는 데 주로 쓰인다. 예컨대 참조범위가 21–27페이지라고 할 때와 같이. 이렇게 쓰인 en-대시의 앞뒤에는 추가 공백을 주지 않고 붙인다.

em-대시나 en-대시는 — 보충적인 주석을 추가하기 위해서 — 문장 일부를 구획하는 문장부호로 쓰이기도 한다. en-대시가 문장부호로 쓰인 경우에는 앞뒤에 스페이스를 둔다. 문제가 되는 것은 em-대시 앞뒤에 어느 정도의 추가 공백을 두느냐는 것인데 이것은 논쟁거리이다. 어떤 토론회에서 참석자의 반 정도는 스페이스 하나 정도의 공백을 넣는 것을 지지하였는데, 나머지 반은 그것을 금기시하였다. em-대시를 사용할 때는 스페이스를 넣든 넣지 않든 일관성있게 사용하는 데 특히 주의하여야 한다.

줄임표는 세개 또는 네개의 점으로 무언가가 줄어졌거나 그 뒤로도 뭔가가 계속된다는 것을 나타내는 문장부호이다. 문장의 중간이나 절이나 ... 등에서 쓰이는데 앞뒤로 공백을 갖는다. 문장의 끝에서는 뒤에 공백을 두지 않고 마침표로 마감하는 것이 영문 문서의 관행인데 결과적으로 네 개의 점을 찍게 된다....

줄표를 몇 글자나 단어 하나 정도를 숨기기 위해 사용하기도 한다. 한 단어 안의 숨겨진 글자들을 나타내기 위해서는 2em-대시(em-대시의 두 배 길이)를 사용한다.

> **snafu,** *(U.S. slang) n.* chaos. — *adj.* chaotic. [*s*ituation *n*ormal — *a*ll *f*——d *u*p.]

3em-대시는 줄어진 단어를 표시하기 위해 사용된다. 내가 메릴랜드에 살 때 우리 동네에서는 *Frederick Post*라는 마을 신문을 발행되고 있었다. 다음 구절은 우연히 읽은 부고기사였는데 그 분의 신원을 보호하기 위해 여기서는 이름을 드러내지 않았다.

> Although he had spent the last 92 years of his life here, Mr. ——— was not a Fredericktonian.

'There's glory for you!'

'I don't know what you mean by "glory" ', Alice said.

Humpty Dumpty smiled contemptuously. 'Of course you don't — till I tell you. I meant "there's a nice knock-down argument for you"!'

'But "glory" doesn't mean "a nice knock-down argument" ', Alice objected.

'When *I* use a word', Humpty Dumpty said, in a rather scornful tone, 'it means just what I choose it to mean — neither more nor less'.

"There's glory for you!"

"I don't know what you mean by 'glory,' " Alice said.

Humpty Dumpty smiled contemptuously. "Of course you don't — till I tell you. I meant 'there's a nice knock-down argument for you!' "

"But 'glory' doesn't mean 'a nice knock-down argument,' " Alice objected.

"When *I* use a word," Humpty Dumpty said, in a rather scornful tone, "it means just what I choose it to mean — neither more nor less."

그림 3.3: 인용 부호: 위쪽은 영국식, 아래쪽은 미국식

3.5 구두점

따옴표

대화의 주변을 둘러싸는 인용부(따옴표)와 관련된 부호들은 수많은 혼선의 원천이다.

미국식 스타일은 인용된 대화에 겹따옴표를 써서 "로 시작하여 "로 끝낸다. 대화 안에 다시 인용되는 말이 있으면 홑따옴표 ('와 ')로 둘러싼다.

영국의 관행은 완전히 반대이다. 대화를 홑따옴표로 둘러싸고 그 안에 다시 인용된 말은 겹따옴표를 쓴다. 어떤 경우든 홑따옴표와 겹따옴표가 붙어나오면 그 사이에 작은 — 단어간격에 해당하는 스페이스는 너무 길고 — 공백을 주어야 구별이 가능할 것이다.

인용문 안에 다시 인용문이 별로 나오지 않는 경우, 내가 보기에 영국식 관행이 미국식에 비하여 얼룩이 덜 지는 것으로 보인다. 그림 (3.3)은 동일한 텍스트를 영국식과 미국식으로 조판한 것이다. 예문은 루이스 캐럴의 「거울나라의 앨리스」에서 따온 것인데 인용문 안의 인용문이 특이할 정도로 많이 사용되고 있다.

따옴표가 올 때 구두점을 어디에다 쳐야 하는가 하는 것도 성가신 문제이다. 이 경우도 미국식과 영국식 관행이 다르다. 미국에서는 쉼표와 마침표를 닫는 따옴표 안에 넣

는 경향이 있고 콜론과 세미콜론은 그 바깥에 둔다. 영국의 편집자들은 쉼표 마침표까지 따옴표의 밖에 내는 것을 좋아한다. 어느 것이 나온지를 결정하기는 어렵다. '올바르다'고 하는 예에 대해서는 반례를 얼마든지 들 수 있다는 느낌이다. 어떤 사람은 이 문제를 회피하기 위해서 아예 쉼표와 마침표를 닫는 따옴표와 같은 위치 아래쪽에 나란히 두기조차 하는데, 이것은 언뜻 보기에 물음표나 느낌표와 혼동될 수 있어 바람직하지 않다. 그림 (3.3)에서 영국식과 미국식 구두점 사용법을 각각 보여주었다. 그러나 두 경우 모두 다 적절해 보이지는 않는 것으로 생각된다. 일관성을 유지하면서 저자의 생각을 잘 전달하는 방법이 어떤 것인지는 기본적으로 문제거리라고 생각한다.

각주 표지

각주 표지를 붙이는 일반적 원칙은 간단하다. 그 주석이 참조하는 텍스트 요소 직후에 붙인다는 것이다. 그러나 여기에는 복잡한 문제가 있다.

참조하는 것이 여러 단어 가운데 있는 한 단어일 경우 이것이 의미하는 것은[1] 명백하다. 문제는 참조하는 부분이 여기 보이는 것과 같이[2], 구(句)일 경우이다. 여기 쉼표는 어느 위치에 와야 하는가. 만약 주석이 언급하는 것이 전체 문장이라면 어떻게 할 것인가?[3]

구두점과 인용부의 경우에도 그랬듯이, 각주표지가 구나 문장의 끝에 있는 마침표와 쉼표의 앞에 와야 하는가 뒤에 와야 하는가? 앞 문단에서 두 가지 위치를 모두 보여주었다.[4] 내가 내린 일반적인 결론은 각주 표지가 구두점의 뒤에 온다는 것이다. 그러나 이런 규칙을 의심하는 사람도 있게 마련이다.

단어에 붙이는 표지로 등록상표 표지와 같은 것도 있다. 이것은 간격을 보기좋게 만들기가 어렵다. 거북함을 없애기 위해서 텍스트를 다시 쓰는 방법밖에는 없을 때도 있다. 앞서 xxiv 쪽에서 한 가지 예가 이미 나왔는데 여기서 나는 'PostScript'가 Adobe Systems 사의 등록상표임을 보이려고 하였다. 다음과 같은 몇 가지 시도를 해보았는데,

> ...languages like PostScript™, presumably ...
>
> ...languages like PostScript®, presumably ...
>
> ...like the PostScript® language, presumably ...

결국 다음과 같이 등록상표를 주석 처리하는 해결책을 채택하고 말았다.

> ...languages, like PostScript[5], presumably ...

[1] 예외가 몇 가지 있기는 하지만

[2] 예를 들기 위해 붙이는 주석이다.

[3] 이 주석은 적절한 위치에 있다고 할 수 있을까?

[4] 각주 2와 3를 보라.

[5] PostScript is a registered trademark of Adobe Systems Incorporated.

이 경우 각주는 'PostScript'라는 단어에만 걸리는 것이 명확하고 따라서 주석 표지를 쉼표의 앞에 두었다.

폰트 변환

이따금 한두 단어가 그 전후 텍스트와는 다른 폰트로 식자되는 수가 있다. 예컨대 강조를 위하여 그 단어를 이탤릭 처리하는 것을 들 수 있다. 이 단어의 뒤에 구두점이 따라오면 본문 글꼴이 아닌 바뀐 글꼴로 구두점을 처리하는 것이 일반적이다. 구두점에 사용된 폰트가 그다지 눈에 띄지 않는 경우도 있으나 어떨 때는 그렇지 않을 수도 있다.

이 문서의 전문에 나오는 *memoir*라는 단어를 볼드체로 식자하도록 하는 방법이 두 종류 있다. 그것은 각각 다음과 같다. 하나는 이어지는 쉼표까지 볼드로 처리한 것이고,

memoir, *n.* ...

다른 하나는 다음과 같다.

memoir, *n.* ...

3.6 좁은 행장

폭이 좁으면 조판하기가 어려워진다. 특히 양끝 맞추기는 대단히 어렵다. 행의 길이가 짧으면 짧을수록 양끝을 맞추기 위해 단어간 간격을 넓게 벌리든가 행끝에서 단어를 잘라도 그 행에 음절을 둘 만한 공간이 없으면 어려움을 겪을 수밖에 없다.

이런 상황에서 가장 좋은 방법은 양끝 맞추기를 포기하고 텍스트를 오른끝 흘림(ragged right)으로 조판하는 것이다. 오른끝 흘림으로 하는 편이 단어 사이에 커다란 공백을 두면서 양끝 맞춤을 하는 것보다 훨씬 낫다. 이 경우 문제가 되는 것은 하이픈처리를 해야 할 것인가 말아야 할 것인가 하는 것이다.

오른끝 흘림의 경우라도 하이픈 처리를 해주면 그러지 않는 경우에 비해 행끝의 들쑥날쑥함이 줄어든다. 행끝을 좀더 가지런해보이게 만들기 위해 하이픈처리를 해줄 수는 있겠지만 짧은 행이 하이픈으로 끝나는 것은 어딘가 이상하다. 이것은 자신의 판단과 디자인 경험에 의존해야 할 것이다.

색인은 2단 편집으로 조판되는 것이 일반적인데 이따금 3단 심지어 4단으로 하는 경우도 있다. 색인은 글줄을 읽어내려가는 것이 아니라 특정 엔트리를 찾아보게 만든 것이다. 그래서 시선의 움직임을 줄여주기 위해 페이지 번호를 색인 항목 바로 뒤에 적어준다. 그 결과 색인은 자연히 오른끝 흘림이 되고 이것이 독자에게 더 편리하다.

하이픈은 언어마다 다른 방식으로 이루어지고 하이픈처리 하는 위치도 다르다. 심지어 영어와 미국어는 거의 차이가 없는 언어를 쓰면서도 하이픈 규칙이 서로 다르다. 개략적으로 말하자면 미국어는 하이픈 위치가 단어의 발음에 더 많이 의존하여 음절 사이에서 끊는 것이 허용된다. 영국식 영어의 하이픈 위치는 단어의 어근에 더 많이 의존하고 있어

서 그 단어가 그리스어 어근이냐 라틴어 어근이냐에 따라 서로 다른 위치에서 끊게 된다. 어떤 단어가 어디에서 하이픈처리할 수 있는 것인지 모를 경우 사전을 찾아보면 끊을 수 있는 위치가 표시되어 있다.

3.7 강조

밑줄긋기 강조방식은 강조해서 말하거니와 사용하지 말아야 한다. 이것은 예전 원고를 타자치던 시절에 별다른 방법이 없어서 쓰던 방식이다. 이밖에 글자 사이를 약간 넓게 띄어서 강조하는 방법이 있는데 이렇게 하면 보통 글자보다 조금 커보일 거라고 생각할지 모르겠지만 사실은 그렇지도 않다.

독일 타이포그라퍼들은 과거 fraktur류의 폰트에서 자간 간격을 넓히는 강조방법을 사용했지만, 오늘날은 폰트 종류와 크기를 이용한 강조 방법을 쓴다.

기본적으로는 세 가지 방법이 있는데, 폰트의 크기를 바꾸는 것, 폰트의 **무게**를 바꾸는 것, 그리고 가장 널리 쓰이는 것으로 폰트의 *shape*를 바꾸는 것 등이다. 무언인가를 강조할 때는 창조적인 긴장이 있다 — 독자에게 강조되는 요소를 분명히 알려주어야 할 필요도 있지만 텍스트의 흐름을 끊고 싶지도 않은 것이다. 세 가지 방법 중에서 shape를 바꾸는 방법이 이 둘 사이의 긴장을 잘 조화시킬 수 있는 합당한 방법이라고 생각한다.

3.8 캡션과 레전드

그림이나 표에 붙이는 제목이라는 뜻의 용어로서 캡션과 레전드가 무엇이 다른지 잘 모르겠다. 그렇지만 레전드는 그림 안에 넣는 설명 문구로서 지도 등의 기호를 설명하는 것을 가리키기도 한다(이것을 '범례'라고 한다).

아무튼 캡션과 레전드는 보통 본문 폰트보다 조금 작은 폰트로 식자한다. 그리고 본문 폰트와 다른 폰트를 쓰기도 한다. 예컨대 본문 폰트가 로만이고 장 표제 폰트가 산세리프 폰트이면 캡션에는 작은 크기의 산세리프 폰트가 적당할 수 있다.

표의 캡션은 표 위에 두고 그림 캡션은 그림 아래 두는 것이 일반적이다.

3.9 표

표라는 것은 텍스트나 숫자가 행과 열로 배열된 것을 말한다. 여기에는 대개 각 열의 상단에 그 열의 데이터가 의미하는 것이 무엇인지를 밝히는 난제(欄題)가 붙는다. 세로 선을 긋거나 괘선을 넣어서 각 난제와 컬럼 엔트리들을 구분한다.

대부분의 타이포그라퍼들은 표에 들어가는 열 구분을 위한 종선을 기피한다. 그 이유를 정확히는 모르겠지만 아마도 한 글자씩 판을 짜야 했던 수공업 조판 시대에 횡선을 짜는 것에 비해서 각 글자들 사이에 적당한 공목을 넣어서 가지런한 세로선을 만들어내

는 것이 상대적으로 상당히 어려웠던 사정이 영향을 미치지 않았을까 싶다. 사람의 눈은 직선이 조금만 흐트러져도 금방 알아내기 때문이다. 전자 조판 시대에 이러한 선의 정렬 문제는 더이상 문제가 되지 않기 때문에 세로선 기피는 아마도 초창기 조판 시대에서 이어받은 전통이 아닐까.

세로선을 사용하려 한다면 그것을 싫어하는 사람도 있다는 것을 알아두도록 하자.

제 4 장

전자책

4.1 들어가는 말

더 좋은 말이 없기 때문에 컴퓨터 화면에서 읽어지도록 만들어진 문서를 전자책 또는 ebook이라고 지칭하겠다. 전자우편 형태의 엄청난 양의 전자문서가 있기는 하지만 여기서 내가 관심을 가지고 있는 것은 몇 분만에 읽을 수 있는 것이 아니라 보다 인쇄본 보고서나 책에 가까운 출판물 형식의 것이다.

이 짧은 절에서는 ebook의 레이아웃에 관한 몇 가지 제안을 하려 한다. 이것은 나의 개인적인 경험에서 비롯된 것이다. 하이퍼링크와 같은 문서 내부 또는 문서 간 찾아가기 장치나 HTML 문서와 같이 모양이 저자에 의해서가 아니라 브라우저나 뷰어에 의해서 달라지는 경우는 문제삼지 않는다.

4.2 고찰

수백년 동안 이어 내려온 진짜 책과는 달리 ebook이 어떻게 만들어져야 하는가에 대한 지침 작성에는 축적된 경험이 거의 없다.

매체가 확연히 다르다. TV 모양의 스크린은 해상도의 한계가 있는 데다가 특정 위치에 고정시켜 두어야 한다. 반면 종이는 접을 수도 있고 표시를 할 수도 있으며 독자가 원하는 곳은 어디든 가지고 갈 수도 있다. 이 차이를 고려하여 다음과 같이 제안할 수 있을 것이다.

책은 얼마든지 휴대가 가능하며 서서 읽을 수도 있다. 컴퓨터 사용자는 의자에 앉아서 책상이나 테이블 위의 모니터를 들아보아야 하고 아니면 노트북으로 읽기를 시도한다. 노트북은 컴퓨터보다는 가볍겠지만 무한정 오랜 시간 들고 있을 수는 없다. 전자책 독자의 물리적 제약을 경감해주기 위해 폰트 크기가 인쇄된 책보다 커야할 것이다. 그러자면 보이는 범위를 더 넓혀야 할 것이다. 폰트가 커지면 자면의 예리함도 증가하므로 각 글자를

디스플레이하는 데 더 많은 픽셀이 필요하다.

내 경험에 의하면 각 페이지를 읽기 위해 화면을 아래 위로 움직이는 스크롤 압박은 대단히 성가신 일이었다. 그러므로 각 페이지는 스크린 안에 맞아 들어가야 한다. 전자책의 페이지는 책의 페이지보다 작아야 한다. 전자책 사이즈로 제안되는 크기는 9×6 인치 [Ado01] 또는 23×15 센티미터이다.

폰트 사이즈는 12포인트 아래여서는 안된다. 폰트는 인쇄에 사용되는 것에 비해서 좀 더 견고한 것이어야 하는데, 미세한 획흘림이나 작은 세리프 따위는 고해상도 스크린이 아니면 잘 표현되지 않는다.

인쇄본 책의 페이지 디자인은 펼침면을 기준으로 한다. 전자책의 디자인은 한 페이지 단위로 이루어진다. 편집영역은 페이지의 중앙에 놓여야 한다. 그렇게 하지 않으면 매 페이지마다 시선이 좌우로 이동하게 되어 귀찮기도 하거니와 곧 지루해진다. 또 헤더와 편집영역의 상단은 스크린 상의 일정 높이에 고정되어 있어야 한다. 페이지의 하단이 일정한 위치를 유지하는 것은 그렇게 심각한 문제가 아니다.

전자책은 종이 책처럼 주르륵 넘기면서 원하는 위치를 찾을 수 없다. 그러므로 찾아가기 장치 — 헤더나 푸터 — 는 더 중요한 문제이다. 각 페이지는 장 표제(절 표제도 있으면 좋다)와 페이지 번호를 꼭 표시해 주어야 한다. HTML 문서에 대해 말하고 있는 것이 아니라는 점을 유념하라.

많은 전자책 리더들은 특정 페이지로 바로 이동하는 기능을 제공한다. 이 때 페이지 번호는 면번호에 나타난 페이지가 아니라 맨 처음부터 붙은 일련번호를 사용한다. 이럴 경우 면번호에는 다른 스타일을 사용하더라도 페이지 번호를 연속해서 붙이는 것이 도움이 될 것이다. 예를 들면 앞부분에는 로마 숫자를 쓰고 본문에는 아라비아 숫자를 쓴다고 했을 때 앞부분의 마지막 페이지가 xi이라면 본문의 첫 페이지를 12로 표시하는 것이다.

전자책에서 빈 페이지가 있어야 할 이유가 없다. 그리고 recto와 verso 페이지에 대한 개념도 적절하지 못하다.

인쇄본 책 중에는 삽화를 접어서 삽입하는 경우가 있다. 이 페이지들은 페이지 번호매김을 하지 않는다. 전자책에서 이러한 삽화는 '전자적으로 삽입'하게 되는데 삽화의 전자적 소스를 삽입하거나 링크를 걸거하 하는 방법을 쓴다. 전자적 형태의 그림을 삽입하는 경우라면 페이지 번호매김에 포함되어야 한다.

인구의 상당 비율이 색맹이라는 사실을 잊지 말라. 보통 적색과 녹색을 구분하는 능력이 떨어지는 경우가 많다. 분홍색 음영이 푸른색으로 인지되거나, 레몬색 오렌지색 라임색이 모두 같은 색으로 인지될 수도 있다. 색맹의 경우 색 기억 능력도 떨어질 수 있다.

색깔을 자유롭게 사용하여 텍스트의 특정 부분이나 그라프 상의 데이터를 나타내는 경우를 전자책에서 자주 본다. 그러나 색깔이 현저히 구별되는 것이 아니라면 독자 중의 약 10% 정도는 그것을 구분하지 못하거나 혼동할 수 있다. 게다가 전자책을 인쇄해서 가지고다니면서 읽으려 할 때 흑백 프린터로 인쇄하면 색상은 음영이나 회색조 처리되어버리므로 구분도 불가능하고 읽을 수도 없게 된다. 흰색 바탕의 노란색, 흰색 위의 더

흰색이나 검정색 위의 네이비블루 등의 색배합을 웹사이트에서 본 적이 있는데 이 페이지를 인쇄했을 때 거의 구분이 불가능하였다.

제 II 부

실전 응용

제 5 장

출발

이 장은 *headings* 페이지 스타일을 사용한다. 페이지스타일에 대해서는 제 14 장을 보라.

일반적으로 memoir 클래스 문서를 작성하려면 `\documentclass[`⟨*options*⟩`]{memoir}` 와 같이 한다. ⟨*options*⟩로 지정할 수 있는 것은 종이 크기, 본문의 활자 크기, 원고의 종류, 수학식 조판상의 지시 등이다.

5.1 용지 크기 옵션

기본 용지 크기는 인쇄할 한 장의 종이의 사이즈를 말한다. 기본 용지 크기 설정 옵션은 다음과 같다.

a3paper 420×297 밀리미터 용지

a4paper 297×210 밀리미터 용지

a5paper 210×148 밀리미터 용지

a6paper 148×105 밀리미터 용지

b3paper 500×353 밀리미터 용지

b4paper 353×250 밀리미터 용지

b5paper 250×176 밀리미터 용지

b6paper 176×125 밀리미터 용지

letterpaper 11×8.5 인치 용지

legalpaper 14×8.5 인치 용지

executivepaper 10.5×7.25 인치 용지

ebook 6×9 인치 크기. 주로 컴퓨터 모니터에서 디스플레이할 '전자책'에 적합한 사이즈이다.

landscape 가로와 세로길이를 바꾸어서 가로가 긴 종이로 조판한다.

landscape만을 제외하고 각 옵션은 상호배타적이므로 함께 쓸 수 없다. 기본값은 letterpaper이다.

5.2 활자 크기 옵션

이 클래스는 상당히 많은 활자 크기 옵션을 제공한다.

9pt for 9pt type

10pt for 10pt type

11pt for 11pt type

12pt for 12t type

14pt for 14pt type

17pt for 17pt type

각 옵션은 상호배타적이고 기본값은 10pt이다.

5.3 인쇄 옵션

인쇄 옵션은 다음과 같다.

twoside 종이의 양면을 모두 이용하여 인쇄할 문서를 작성한다.

oneside 단면 인쇄용 문서를 작성한다.

twoside와 oneside 옵션은 상호배타적이므로 동시에 사용할 수 없다.

onecolumn 한 페이지에 텍스트를 1단으로 배열한다.

twocolumn 같은 너비(width)를 가지는 2단 문서를 작성한다.

onecolumn과 twocolumn 옵션은 상호배타적이다.

openright 장(chapter)이 시작하는 페이지를 홀수쪽(펼침면의 오른쪽)으로 한다.

openleft 장(chapter)이 시작하는 페이지를 짝수쪽(펼침면의 왼쪽)으로 한다.

openany 장(chapter)을 짝수·홀수쪽 어디에서나 시작하도록 한다.

openright, openleft, openany 옵션은 상호배타적이다.

final 작성한 문서의 인쇄용 최종본을 만든다.

draft 행넘침(overfull)이 있을 때 검은 선을 그어주고, 몇 가지 변화점을 표시해준다. 어떤 패키지를 사용하면 다른 표시나 효과도 발생한다.

ms 이 옵션은 문서가 타자로 친 것과 비슷하게 보이도록 한다. 어떤 출판편집자는 이렇게 초라하게 보이는 원고로 제출해야 좋아하는 경우가 있다.

final, draft, ms 옵션은 상호배타적이다.

showtrims 이 옵션은 최종본을 인쇄한 다음 종이의 어느 부분을 잘라내야 할 것인지를 표시해주는 트림 마크를 찍어준다.

기본값은 twoside, onecolumn, openright, final이다.

5.4 기타 옵션

그밖의 옵션은 다음과 같다.

leqno 수식(equation) 번호가 왼쪽에 붙는다(기본값은 오른쪽).

fleqn 수학표현식이 왼쪽 마진에서 `\mathindent` 만큼 들여쓰기되는 문단으로 조판된다(기본값은 중앙정렬).

openbib 참고문헌(bibliography) 항목이 각각 새로운 줄로 식자되고 두번째 줄부터 `\bibindent` 만큼 들여쓰기된다(기본값은 들여쓰기 없이 계속 이어지는 것).

article article 클래스를 흉내내도록 한다. 그러나 `\chapter` 명령은 여전히 효력이 있으며 각 장(chapter)은 새로운 페이지에서 시작하지 않는다. chapter headings는 section heading인 것처럼 식자된다. 그림 등의 번호는 장이 바뀌어도 재설정되지 않고 일련번호로 매겨진다. 그러나 `\part` 명령은 그 자체가 한 페이지로 조판된다.

oldfontcommands 예전의 LaTeX version 2.09 폰트 명령을 사용할 수 있게 한다. old font 명령을 사용할 때 경고 메시지를 만날 수 있다.

이 옵션은 모두 기본값이 아니다.

5.5 附記

이 클래스를 아무런 옵션도 지정하지 않고 부르는 것은 다음과 동일하다:

```
\documentclass[letterpaper,10pt,twoside,onecolumn,openright,final]{memoir}
```

이 사용자 설명서는 다음과 같이 시작하였다.

```
\documentclass[letterpaper,10pt]{memoir}
```

letterpaper나 10pt가 모두 기본값이므로 이들은 쓰지 않아도 상관없다.

문서의 실제 부분은 document 환경 안에 들어간다. \documentclass 명령에서부터 document 환경까지 영역을 프리앰블이라 한다. 이 영역에서는 외부 패키지를 부르거나 자신의 매크로를 정의한다.

```
\flushbottom \raggedbottom
```

twoside나 twocolumn 옵션이 주어지면 문서는 \flushbottom으로 식자된다. 그외의 경우는 \raggedbottom이다.

\raggedbottom 선언은 LaTeX이 조판 영역의 높이를 일정하게 유지하려는 시도를 하지 않아도 되도록 하는 효과가 있으므로 페이지가 조금 부족하게 채워지는 경우도 생긴다.

\flushbottom 선언은 LaTeX이 각 페이지의 조판 영역을 되도록 일정한 높이로 유지하도록 한다. 페이지 나눔 명령이 의도적으로 쓰인 경우를 제외하고. 페이지의 높이를 유지하는 방법은 수직 간격(예를 들면, 문단과 문단 사이, 장절명령의 전후 공간, 떠다니는 개체의 전후공간 등)을 늘리거나 줄이는 것이다.

\raggedbottom과 \flushbottom을 문장의 중간에 써서 페이지를 조절할 수도 있다.

그러나 일반적으로 프리앰블에 \raggedbottom을 선언해두는 것이 좋을 것이다.

ebook 옵션을 쓸 때는 12pt와 oneside 옵션을 함께 쓰는 것도 좋다.

5.6 한글 사용

【memhangul】 이 절은 번역본에서 추가한 것이다. 번역본에서 한글화와 관련된 부분은 이 문단과 같이 \small 사이즈의 고딕체로 표시되어 있다. ■

memoir 클래스에서 한글을 사용하기 위한 목적으로 만들어진 것이 memhangul-ucs이다. 처음에 한글화는 HLaTeX의 hfont를 이용하여 시도되었고 그 스타일 이름이 memhangul이었다. 나중에 memhangul은 dhucs를 사용하는 memhangul-ucs로 발전하였다. 현재는 memhangul은 더이상 지원되지 않는다.

그러므로 한글 사용을 위해서 다음 문장을 프리앰블에 포함한다.

`\usepackage[`⟨*options*⟩`]{memhangul-ucs}`

memhangul-ucs는 유니코드 한글을 사용한다. 모든 입력파일의 한글은 UTF-8 인코딩된 유니코드로 표기되어야 한다.

스타일 옵션은 다음과 같다.

interwordHWP 한글 자간과 단어간격을 HWP에서와 유사하게 설정한다.

interworddefault 한글 자간과 단어간격을 default로 설정한다.

interwordHWP와 interworddefault는 상호배타적이다.

nosetspace 행간격 자동 설정 기능을 끈다. 이 옵션이 주어지면 영문판 setspace를 로드한 것과 효과가 같으므로 `\SetHangulspace`와 같은 명령을 사용할 수 없다. 이 경우에도 행간을 1.333으로 맞추는 것이 기본값이다.

noquotespacing `quote, quotation`의 행간격 재설정 기능을 끈다. 필요하다면 선언형으로도 사용할 수 있다.

gremph `\emph` 명령에 의해서 식자되는 부분을 기울임으로 하지 않고 그래픽 글꼴을 쓰도록 한다. `\hangulemphtrue`, `\hangulemphfalse` 명령으로 문서 중간에 재설정할 수도 있다.

nonfrench 한글에 nonfrenchspacing을 적용한다. 이 옵션을 준 경우에는 `\xspaceskip` 값을 설정할 수 있다.

memhangul-ucs를 이용하여 한글 문서를 작성하는 보다 구체적인 예로는 `memucstest.tex`이라는 문서의 소스를 참고하라. 한글화를 위하여 상당히 많은 기능이 추가되었지만 이 글에서 그것을 모두 설명하지는 못하였다. 별도의 매뉴얼이 필요하리라 생각한다.

제 6 장

페이지 레이아웃

이 장은 *ruled* 페이지스타일로 조판한다.

6.1 들어가는 말

이 클래스의 기본 페이지 레이아웃은 다음과 같다. 페이지 크기(판형)는 용지 크기와 같고 편집 영역(판면)은 페이지의 얼추 가운데 부분에 온다. 이 장에서는 이러한 기본 페이지 레이아웃에 만족하지 못하는 경우 자신의 페이지 레이아웃을 만드는 명령들을 설명한다.

책의 페이지가 잘 설정되어야 독자를 설득하고 즐겁해하고 매혹할 수 있다. 페이지 디자인은 저자의 의도를 전달하고 독자가 저자의 생각을 따라잡을 수 있도록 설계되어야 한다. 일반적인 독자가 페이지 레이아웃을 의식하지 않는 것이 좋은 페이지 디자인이다. 만약 무의식적으로라도 끊임없이 페이지 레이아웃이 신경쓰인다면 책의 원래 목적을 벗어나게 될 것이다. 디자이너가 자신의 작품을 독자에게 보여주기 위해서 발악하거나 "이것 좀 봐줘"라고 중얼거리는 디자인은 좋지 않다.

판형(페이지)에는 세 가지 주요 구성요소가 있다. 페이지 자체, 판면(편집 영역), 그리고 여백(margin)이 그것이다. 여백은 페이지의 끝(edge)에서 편집 영역을 분리시키는 영역이다. 그리고 이보다 조금 덜 중요한 것으로 상단면주(머릿글, header), 하단면주(바닥글,footer), 여백문단(marginalia)이 있다. 페이지 디자인의 기법은 이 요소들이 조화로운 균형과 리듬을 갖도록 하는 것이다.

형식과 명령은 다르지만 이 장에서 설명하는 기능들은 geometry 패키지가 하는 일과 유사하다.

6.2 용지

인쇄라는 것은 기호들을 어떤 고정 물체 위에 찍는 행위이다. 실크 스크린으로 T셔츠에

인쇄하는 경우, 기호들은 망(screen)에 새겨지고, 그 다음 잉크액이 망을 투과하여 고정물체 — 이 경우에는 T셔츠 직물 — 위로 압출된다. 이 과정에 보통 사람들이 흥미를 가지거나 그렇지않거나 간에, 책을 만드는 것은 일반적으로 이와는 다른 종류의 대상물에 다른 방법으로 인쇄한다. 아주 특별한 경우를 제외하고 책은 종이에 인쇄하는 것이다.

탁상출판의 경우, 일련의 표준 치수의 용지가 일반적으로 쓰인다. 미국에서는 letter-paper (11×8.5 인치)가, 다른 곳에서는 A4 규격 (297×210 mm)이 흔히 쓰이는 사이즈이다. 이 용지 위에 한 페이지씩 출력하는 것이 일반적이다. 상업 출판에서는 여러 페이지가 한 장에 인쇄되는 훨씬 큰 사이즈의 용지가 사용된다. 이 큰 지면에 여러 페이지를 인쇄하여 접고(접지) 나누고 재단하고 묶어서 책을 만든다. 우리는 탁상출판의 경우만을 생각하기로 한다.

6.3 판형(페이지)

우리는 한 장에 한 면만 인쇄되는 경우를 생각한다.

LaTeX이 페이지 레이아웃 설정에 사용하는 인자(parameter)들을 그림 (6.1) 에 보였다. LaTeX은 용지의 실제 크기에는 상관하지 않고, (top:left)를 기준으로 종이가 가로 · 세로로 무한하다고 가정한다. 편집 영역이 인쇄될 종이보다 너무 넓거나 길다면 LaTeX은 간단히 실제 경계보다 바깥쪽에 텍스트를 찍는다.

LaTeX 파라미터들은 그다지 편리하지 않다. 예를 들어 텍스트의 상단에 일정한 거리를 두고 배쪽(foredge) 마진이 등쪽(spine) 마진의 두 배가 되어야 한다면, 각 파라미터 값을 다시 계산해야 한다. 피곤한 일이다.

이 클래스는 그보다 좀 편리하게 페이지 레이아웃을 조절할 수단을 제공한다. 조절 가능한 파라미터들을 이용해서 용지 크기, 판형 크기 등을 설정할 수 있도록 한다. 그림 (6.2)는 홀수쪽의 판형을 주요 레이아웃 파라미터와 함께 그려서 보여주고 있다. 이 각각은 `\setlength` 명령 또는 아래 설명할 명령을 이용하여 개별적으로 바꿀 수 있다.

표준 클래스 코드에는 다음과 같이 되어 있다.

> '`\paperwidth`와 `\paperheight`는 재단 후의 실제 종이 크기를 나타낸다. 탁상 인쇄기에서 이것은 용지 크기와 같다. 왜냐하면 인쇄 후처리가 없기 때문이다. 실제 책 제작을 위한 클래스에서는 용지와 판형 크기를 다르게 하고 재단선 표지(crop mark)를 넣어주어야 할 것이다.'

이 클래스는 `\stockwidth`와 `\stockheight` 길이를 추가하여, 각각 재단 전 실제 용지 크기를 나타내도록 하였다.

페이지 레이아웃 디자인의 첫 단계는 판형 크기를 결정하고 적당한 인쇄 용지 사이즈를 선택하는 일이다. 표준 용지를 선택하는 것이 간편하다. 별도로 설계된 용지를 이용하려면 보다 주의를 기울여야 한다.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

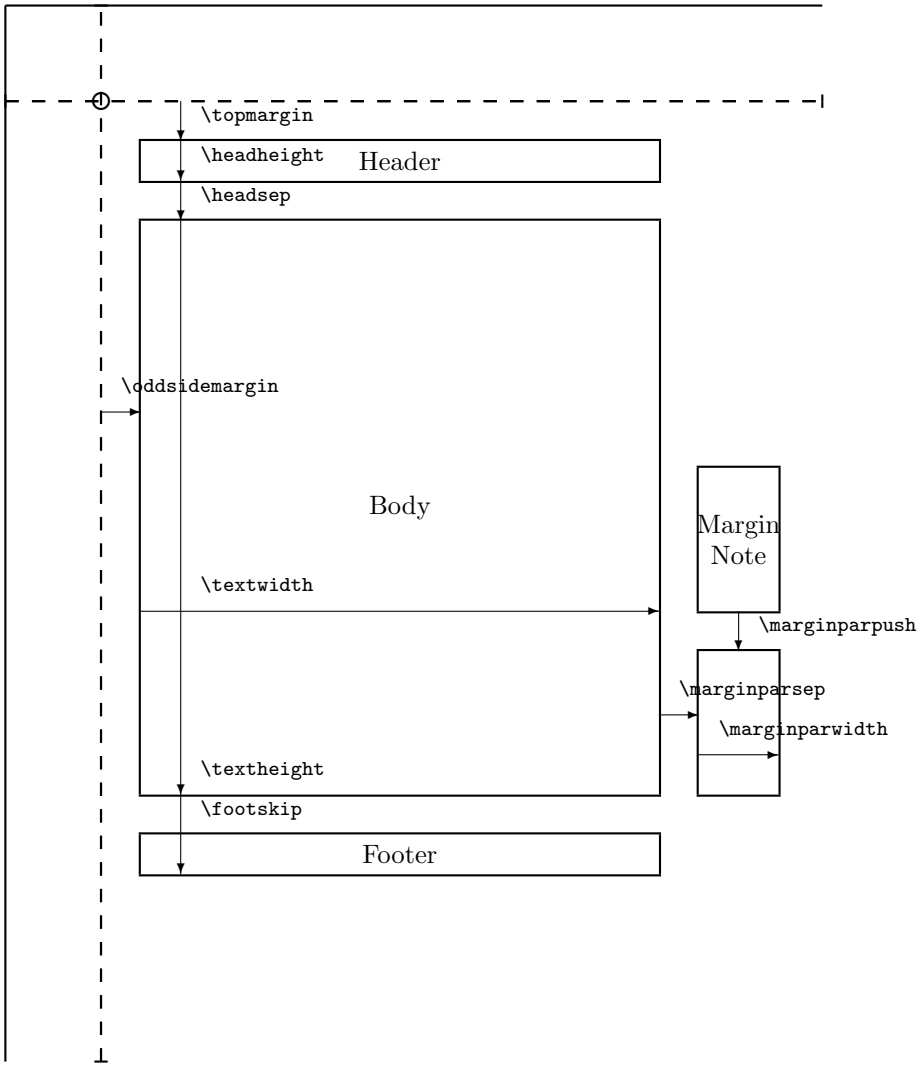


그림 6.1: LaTeX 페이지 레이아웃 홀수쪽 파라미터

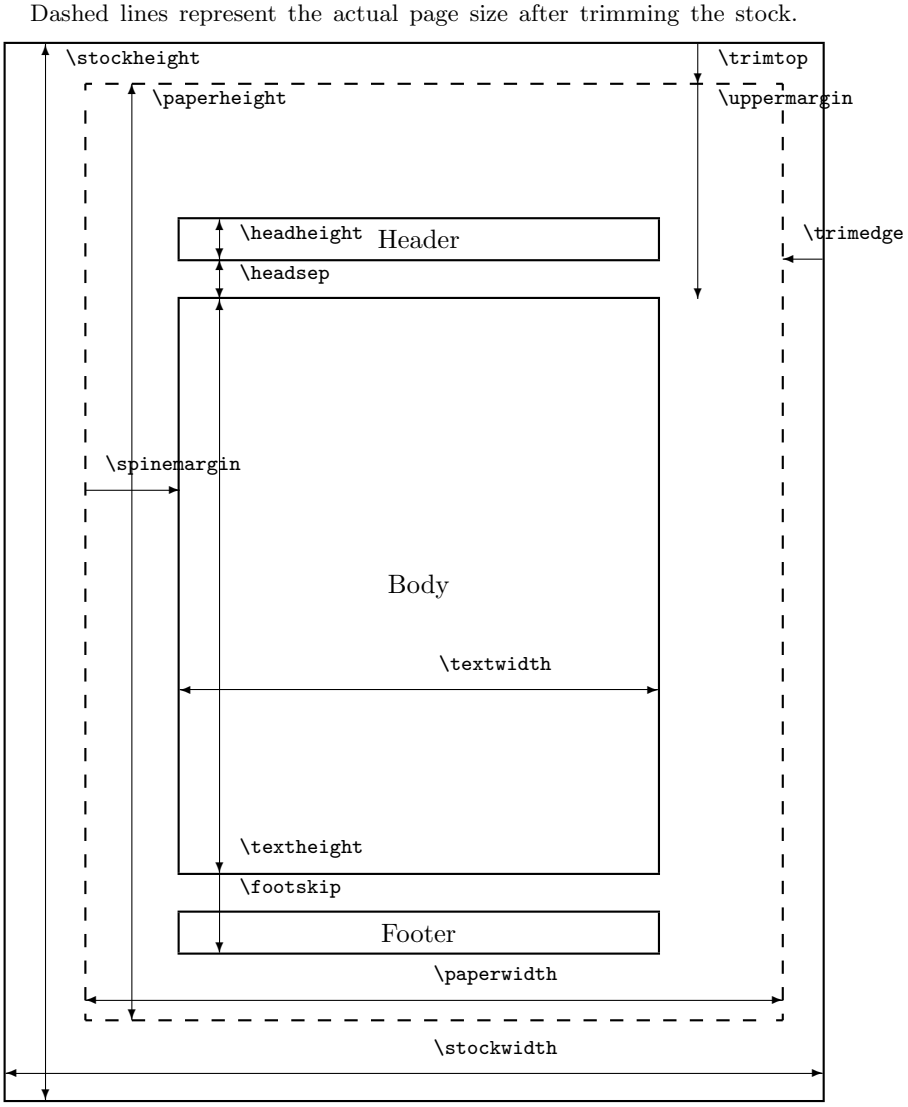


그림 6.2: memoir 클래스의 주요 페이지 레이아웃 파라미터, 홀수쪽.

\setstocksize{⟨*height*⟩}{⟨*width*⟩} \stockheight \stockwidth

이 클래스는 몇 가지 표준 용지 사이즈를 제공한다. 표준이 아닌 별도 사이즈를 설정하려면 \setstocksize 명령을 쓴다. 인자는 ⟨*height*⟩ × ⟨*width*⟩로 주어진다. 예를 들어, 다음 명령은 용지를 9 × 4 인치 크기로 설정한다.

```
\setstocksize{9in}{4in}
```

판형 크기는 용지 크기보다 커서는 안된다. 그러나 인쇄 후에 재단을 거쳐서 판형 크기로 줄어질 것이므로, 더 작을 수는 있다. 판형은 용지의 어느 위치에든 놓일 수 있다.

페이지 레이아웃은 항상 펼침면(spread)을 염두에 두어야 한다. 책을 폈을 때 왼쪽에는 짝수쪽이, 오른쪽에는 홀수쪽이 오고 가운데는 접힘자리(책등)가 있다. 서적은 접었을 때 가로 길이보다 세로 길이가 길다. 그래야 펼쳐 들고 읽기 수월할 것이다. 가로 길이가 매우 긴 페이지를 여러 번 접어서 끼워넣은 '접은 페이지'는 특별히 표 따위를 위해서 넣는 것이 아닌 한 피해야 한다.

\settrimmedsize{⟨*height*⟩}{⟨*width*⟩}{⟨*ratio*⟩} \paperheight \paperwidth

\settrimmedsize 명령은 재단 후의 판형 높이와 폭을 지정하기 위해 사용한다. 애초 판형 크기는 paper size 옵션에 의해 설정된 용지 사이즈와 동일하게 맞추어져 있다. ⟨*ratio*⟩ 인자는 ⟨*height*⟩나 ⟨*width*⟩에 일정 비율을 곱해서 width 또는 height 값을 설정하도록 할 때 쓴다. 세 개의 인자 가운데 두 개의 인자만을 실제값으로 주어야 하며, 다른 하나의 값이 주어지지 않은 인자는 *로 표시한다. \paperheight와 \paperwidth는 주어진 인자에 따라 계산된다. 즉, \paperheight와 \paperwidth를 직접 지정하거나 둘 가운데 하나의 값을 준 다음 곱하는 수를 ratio로 지시할 수 있는 것이다.

\setstocksize로 용지를 재설정하였을 때, 판형을 동일하게 하려면 다음과 같이 하면 된다.

```
\settrimmedsize{\stockheight}{\stockwidth}{*}
```

또는, 용지의 90%를 페이지 사이즈로 사용하려면:

```
\settrimmedsize{0.9\stockheight}{0.9\stockwidth}{*}
```

다음은 8×5 인치 판형을 정의하는 세 가지 다른 방법들이다.

```
\settrimmedsize{8in}{5in}{*}
\settrimmedsize{8in}{*}{0.625}  % 5 = 0.625 times 8
\settrimmedsize{*}{5in}{1.6}    % 8 = 1.6 times 5
```

잘 만들어진 하드카버 책을 보면 낱장들이 책등(spine)을 중심으로 서로 이어져 있는 것을 볼 수 있다. 책을 책장에 똑바로 꽂아두었을 때, 먼지가 윗쪽에서 책장 사이로 스며들기 어렵도록 상단을 부드럽게 처리하여야 한다. 그러므로, 용지를 재단할 때 상단을 잘라낸다. 페이지의 바깥쪽(배쪽, foredge)과 바닥도 잘려나간다. 그러지 않으면 책을 펴거나 읽을 수 없을 것이니까.

\settrims{⟨*top*⟩}{⟨*edge*⟩}
\trimtop \trimedge

`\settrims` 명령은 용지의 상단과(⟨*top*⟩)과 바깥쪽(⟨*edge*⟩)을 재단할 분량을 지시한다. `\settrims`와 `\settrimmedsize`를 결합하면 판형을 용지 상의 원하는 위치에 놓을 수 있다. top과 edge 재단값은 0이 기본값이다. 이것은, 재단을 해야 한다면 바닥과 책등쪽(오른쪽 페이지의 왼쪽)을 자르는 것이 기본값이라는 뜻이다.

재단값은 직접 특정치를 지정할 수도 있고 LaTeX 계산으로 할 수도 있다. 예를 들어, 가로 5인치, 세로 8인치 판형을 가로 10인치 세로 7인치 용지상에 설정하려 할 때,

```
\settrims{2in}{2in}
```

위와 같이 지정하면 원하는 판형의 영역이 상단과 바깥쪽(foredge)에서 2인치씩을 잘라내도록 지정하는 것이다. 예컨대 페이지를 용지의 90% 가 되게 하는 등, LaTeX 계산이 필요하다면, 다음과 같이 한다.

```
\setlength{\trimtop}{\stockheight}    % \trimtop = \stockheight
\addtolengh{\trimtop}{-\paperheight}  % \trimtop = \stockheight - \paperheight
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

위의 설정은 잘라낼 치수를 모두 상단과 바깥쪽 끝으로 가도록 설정한 것이다. 원한다면 다음과 같이 설정하여 상하단의 절단값을 동일하게 할 수 있다.

```
\settrims{0.5\trimtop}{\trimedge}
```

6.4 판면(편집 영역)

판형과 마찬가지로, 판면도 가로보다 세로가 긴 사각형으로 이루어진다.

표 [6.1]은 Computer Modern Roman 글꼴 소문자의 글꼴배수를 보여준다. 이것과 29 쪽의 표 [2.2]을 연결시키면 적절한 textwidth를 결정할 수 있다.

\xlvchars \lxvchars

표 6.1: CMR 글꼴 소문자 알파벳의 길이

Font size	Length (points)
5 pt	87
6 pt	94
7 pt	102
8 pt	108
9 pt	118
10 pt	128
11 pt	139
12 pt	150
14 pt	175
17 pt	207
20 pt	245
25 pt	290

이 표에 의하여, \xlvchars와 \lxvchars라는 두 개의 길이 변수는 문서의 글꼴 크기에 따라 각각 45 또는 65글자 한 행의 길이와 대략 같도록 정해진다.

다른 폰트의 경우라면 다음과 같이 하여 필요한 길이를 얻을 수 있다.

```
\newlength{\mylen}                % a length
\newcommand{\alphabet}{abc...xyz} % the lowercase alphabet
\begingroup                       % keep font change local
% font specification e.g., \Large\sffamily
\settowidth{\mylen}{\alphabet}
The length of this alphabet is \the\mylen.  % print in document
\typeout{The length of the Large sans alphabet is \the\mylen} % in log file
\endgroup       % end the grouping
```

\typeout 매크로는 이 결과를 터미널과 로그 파일에 기록한다.

Morton Høgholm은 표 [2.2]의 데이터를 근사추정하여 다음 관계식을 발견했다.

$$L_{65} = 2.042\alpha + 33.41\text{pt}$$

그리고

$$L_{45} = 1.415\alpha + 23.03\text{pt}$$

여기서 α는 알파벳의 길이이고 L_i는 i개 문자를 포함한 행 길이를 나타낸다. 그래서 그는 다음 매크로를 제안하였다.

\setlxvchars[⟨*fontspec*⟩]
\setxlvchars[⟨*fontspec*⟩]

setlxvchars와 \setxlvchars는 각각 ⟨*fontspec*⟩ 폰트의 \lxvchars와 \xlvchars 길이를 설정한다.

예를 들어 다음 명령이 불린 후에 \lxvchars와 \xlvchars 값은,

```
\setlxvchars \setxlvchars[\small\sffamily]
```

각각 \lxvchars = 293.93684pt, \xlvchars = 179.56801pt가 된다.

이 주제와 관련해서, Morten은 다음과 같이 쓰고 있다.

> ... 나는 \parindent 값을 들여밀기로 하는 환경을 몇 가지 정의해보았다. 몇 가지 이유에서 \parindent 대신 1.5em이라고 코딩하였는데, 이 두 값이 같은 것인 줄 알았기 대문이다. 그런데 mathpazo 패키지를 로드하였더니 여러 가지 \fontdimen이 변경되는 것을 볼 수 있었다. 결국, 내가 정의한 환경은 \parindent가 17.6207 pt인데도 여전히 1.5 em = 18.0 pt를 집어넣고 있었다.
>
> 이런 상황을 피하려면 다음 한 줄을 \documentclass 앞에 써주면 된다.
>
> \RequirePackage{⟨*font-package*⟩}\normalfont
>
> 그러나 나는 이런 제안을 한 번도 본 적이 없다. 대부분의 문서 클래스는 현재 글꼴 설정을 따르고 있을 것이고 이를테면 \parindent 등에 대해 같은 일을 겪고 있을 것이다.

판면의 높이(height)는 행의 수에 대응한다.

\settypeblocksize{⟨*height*⟩}{⟨*width*⟩}{⟨*ratio*⟩}
\textheight \textwidth

\settypeblocksize 명령은 판면의 \textheight와 \textwidth를 설정한다는 점만 제외하면 \settrimmedsize와 같다. 예를 들면 판면을 가로 3인치, 세로 6인치로 설정하는 데는 다음과 같은 세 가지 방법이 있다.

```
\settypeblocksize{6in}{3in}{*}
\settypeblocksize{6in}{*}{0.5}
\settypeblocksize{*}{3in}{2}
```

판면은 판형 위에 놓인다. 판형과 판면과 여백 사이에는 일정한 관계가 있다. 등쪽(안쪽) 여백과 배쪽(바깥쪽) 여백, 그리고 판면의 폭을 더하면 판형의 폭과 같아야 한다. 마찬가지로, 상단 여백, 하단 여백과 판면의 높이를 합하면 판형의 높이와 같다. 판면을 판형 위에 놓는 것은, 판형의 네 귀퉁이에 대해서 판면의 (상대적) 위치를 잡는 방법으로 생각해볼 수도 있고, 판면과 판형 페이지 사이에 여백 값을 설정하는 방식으로 접근할 수도 있다.

표 6.2: \setlrmargins의 인자와 결과

⟨*spine*⟩	⟨*edge*⟩	⟨*ratio*⟩	Result
S	E	r	이상함(ambiguous)
S	E	*	이상함(ambiguous)
S	*	r	이상함(ambiguous)
S	*	*	$E = K_w - S$
*	E	r	이상함(ambiguous)
*	E	*	$S = K_w - E$
*	*	r	$E + S = K_w,\ E = rS$
*	*	*	$E + S = K_w,\ E = S$

페이지 레이아웃은 펼침면 모양을 기준으로 정의된다는 점을 잊어서는 안된다. 배쪽 여백은 등쪽 여백의 반 정도가 일반적이다. 그 이유는 여백 부분이 텍스트의 주변을 고르게 에워싸도록 하기 위해서이다.

상단과 하단 여백의 비율을 결정하는 것은 좀더 자유롭다. 대개 위쪽 마진을 아래쪽 마진보다 조금 적게 잡는 경우가 많아서, 판면이 판형 페이지의 수직 중앙에 오지 않는다.

판형의 가로 길이(폭)를 설정하는 두 가지 방법을 제공한다. 하나는 판면의 폭이 고정되어 있고 여백이 조절가능하다고 보는 것이고, 다른 하나는 여백 크기가 고정되어 있고 판면의 폭이 조절가능하다고 보는 것이다.

\setlrmargins{⟨*spine*⟩}{⟨*edge*⟩}{⟨*ratio*⟩}
\spinemargin \foremargin

\setlrmargins 명령은, 판형과 판면이 고정되어 있다고 보고 좌우 여백을 설정한다[1]. 이 때, 인자 값은 하나만 주거나 아예 주지 않아야 한다. 둘 이상의 값을 동시에 설정하면 안된다. 값이 주어지지 않은 인자는 *(애스테리스크)로 표시한다. 표 [6.2]에 고려해야 할 몇 가지 경우를 표로 나타내었다.

표에서, S는 안쪽 마진 계산값이고, E는 바깥쪽 마진 계산값이다. P_w와 B_w는 각각 페이지와 편집 영역의 폭이다. \setlrmargins 명령은 다음 관계식을 충족한다.

$$S + E = K_w = \text{상수}\ (= P_w - B_w).$$

"이상함"이라고 표시된 것은 인수값들의 조합이 모두 충족되면 위의 관계식을 만족하지 못할 수 있는 경우를 말한다.

예를 들어, 5인치 페이지에 3인치 판면을 사용하고, 등쪽 여백을 0.8인치, 배쪽 여백을 1.2인치(즉, 바깥쪽 마진은 안쪽 마진의 반만큼 더 크다)로 설정하려 한다. 다음 세 가지

[1]그림 (6.2)에는 안쪽 마진만이 표시되어 있다. 바깥쪽 마진은 편집영역 건너편의 반대쪽에 있는 것이다.

표 6.3: \setlrmarginsandblock의 인자와 결과

⟨*spine*⟩	⟨*edge*⟩	⟨*ratio*⟩	Result
S	E	r	S, E
S	E	*	S, E
S	*	r	$E = rS$
S	*	*	$E = S$
*	E	r	$S = rE$
*	E	*	$S = E$
*	*	r	ambiguous
*	*	*	ambiguous

방법으로 이와 같은 효과를 얻을 수 있다. 다만 \pagewidth와 \textwidth는 이미 주어진 고정값이다.

```
\setlrmargins{0.8in}{*}{*}   % specify spine margin
\setlrmargins{*}{1.2in}{*}   % specify foredge margin
\setlrmargins{*}{*}{1.5}     % specify foredge/spine ratio
```

\setlrmarginsandblock{⟨*spine*⟩}{⟨*edge*⟩}{⟨*ratio*⟩}

\setlrmarginsandblock 명령은, 등쪽·배쪽 여백을 설정하는 데 쓴다. 이 때 판형의 폭은 주어져 있고, 판면의 폭은 여백에 따라 달라지는 것으로 생각한다. 이 명령이 주어진 경우의 결과를 표 [6.3]에서 볼 수 있다. 이 경우 \setlrmarginsandblock 명령은 다음 관계식을 만족한다.

$$S + B_w + E = \text{상수 } (= P_w).$$

판면의 가로길이(폭)는 $B_w = P_w - S - E$로부터 계산된다.

판면의 폭이 3인치, 판형 폭이 5인치, 그리고 등쪽 여백은 0.8인치, 배쪽 여백은 1.2인치(즉, 바깥쪽 마진은 안쪽 마진보다 1.5배 크다)로 설정하려 하는 경우, 다음과 같이 할 수 있다. 단, \textwidth는 이미 주어진 \pagewidth와 여백 길이값들로부터 계산된다.

```
\setlrmarginsandblock{0.8in}{1.2in}{*}   % specify both margins
\setlrmarginsandblock{0.8in}{*}{1.5}     % specify spine & foredge/spine ratio
\setlrmarginsandblock{*}{1.2in}{0.667}   % specify foredge & spine/foredge ratio
```

만약 여백을 모두 1인치로 하고 싶으면 다음과 같이 하면 된다.

```
\setlrmarginsandblock{1in}{1in}{*} % specify both margins
\setlrmarginsandblock{1in}{*}{1}   % specify spine & foredge/spine ratio
\setlrmarginsandblock{1in}{*}{*}   % specify spine (foredge = spine)
\setlrmarginsandblock{*}{1in}{1}   % specify foredge & spine/foredge ratio
```

표 6.4: \setulmargins의 인자와 결과

⟨*upper*⟩	⟨*lower*⟩	⟨*ratio*⟩	Result
U	L	r	ambiguous
U	L	*	ambiguous
U	*	r	ambiguous
U	*	*	$L = K_h - U$
*	L	r	ambiguous
*	L	*	$U = K_h - L$
*	*	r	$L + U = K_h,\ L = rU$
*	*	*	$L + U = K_h,\ L = U$

```
\setlrmarginsandblock{*}{1in}{*}   % specify foredge (spine = foredge)
```

여기까지가 폭(수평 길이)을 설정하는 방법이다. 이제부터, 높이(세로 길이)를 설정하는 명령에 대해 알아보자.

\setulmargins{⟨*upper*⟩}{⟨*lower*⟩}{⟨*ratio*⟩}
\uppermargin \lowermargin

\setulmargins 명령은 판형의 높이(세로 길이)와 판면의 높이가 고정되어 있다고 보고 상단 하단 여백을 설정한다[2]. 이것은 \setlrmargins의 경우와 비슷하다. 기호가 약간 다른데, U가 상단 여백, L이 하단 여백이고 P_h와 B_h는 판형과 판면의 높이를 각각 나타낸다. 이 명령의 인자와 결과는 표 [6.4]에 나와 있다. \setlrmargins 명령은 다음 관계를 만족한다.

$$U + L = K_h = \text{상수 } (= P_h - B_h).$$

\setulmarginsandblock{⟨*upper*⟩}{⟨*lower*⟩}{⟨*ratio*⟩}

\setulmarginsandblock 명령은, 판형 높이가 주어져 있고 판면의 세로 길이는 여백에 좌우된다는 조건 하에서 상단 하단 여백을 설정하는 데 쓰인다. 이 명령이 가져오는 결과는 표 [6.5]에서 볼 수 있다. \setulmarginsandblock 명령은 다음 관계식을 만족한다.

$$U + B_h + L = \text{constant } (P_h).$$

판면의 높이는 $B_h = P_h - U - L$로부터 얻어진다.

[2]그림 (6.2)에는 상단 여백만이 표시되어 있다. 하단 여백은 판면 바닥에서 페이지 아래쪽 끝까지의 길이이다.

표 6.5: \setulmarginsandblock의 인자와 결과

⟨*upper*⟩	⟨*lower*⟩	⟨*ratio*⟩	Result
U	L	r	U, L
U	L	*	U, L
U	*	r	$L = rU$
U	*	*	$L = U$
*	L	r	$U = rL$
*	L	*	$U = L$
*	*	r	ambiguous
*	*	*	ambiguous

```
\setcolsepandrule{⟨colsep⟩}{⟨thickness⟩}
\columnsep \columseprule
```

twocolumn 텍스트에서는 단분리 영역의 폭을 지정해야 한다. 단분리 영역에 수직선을 그을 수도 있다. \setcolsepandrule 명령은 단분리 영역의 폭 \columnsep을 인자 ⟨*colsep*⟩으로 설정하고, 단분리 선의 두께를 ⟨*thickness*⟩로 지정하는 데 쓰인다. ⟨*thickness*⟩ 인자가 0pt이면 선을 긋지 않는다. 일반적으로 이 선의 두께는 0.4pt이다. twocolumn 텍스트의 폭과 단분리 영역의 폭을 합치면 편집 영역의 폭과 같아져야 한다.

이리하여, 판형 크기, 판면 크기 및 판면을 둘러싸는 여백의 크기라는, 페이지의 기본 구성요소를 설정하였다.

6.5 면주와 여백 문단

페이지에는 두 가지 추가적인 요소가 있다. 일반적으로 그 가운데 하나 정도는 반드시 존재한다. 이들을 상단 면주(running header)와 하단 면주(running footer)라 한다. 페이지 번호를 붙일 때는 상단이나 하단의 면주 위치에 온다. 실제로 페이지 번호가 반드시 면주 영역에만 오는 것은 아니지만 항상 어딘가에는 오게 된다. 페이지 번호가 붙는 일반적인 위치는 페이지의 바깥쪽 상단 면주나 하단 면주이다. 그래야 책 전체를 대강 훑어보기 쉬워지기 때문이다. 하단의 가운데 오는 경우도 있다. 그러나 독자를 특별히 괴롭히고 싶은 경우가 아니라면 책 안쪽에 바짝 들여 놓지는 말아야 한다.

보통 페이지 머릿글에는 한쪽에 현재 장의 제목을, 다른 쪽에 절의 제목을 넣어주는 경우가 많다. 독자로 하여금 쉽게 찾아볼 수 있도록 하기 위해서이다. 어떤 책에는 책 제목을 짝수쪽 상단 면주에 넣는 경우가 있는데, 독자는 책 제목을 이미 알고 있을 것이기 때문에 이렇게 하는 것이 무슨 도움이 되는지 모르겠다. 차라리 이 공간을 좀더 유용한 정보를 제공하도록 하는 것이 나을 것이다.

표 6.6: \setheaderspaces의 인자와 결과

⟨*headdrop*⟩	⟨*headsep*⟩	⟨*ratio*⟩	Result
D	H_s	r	ambiguous
D	H_s	*	ambiguous
D	*	r	ambiguous
D	*	*	$H_s = C_h - D$
*	H_s	r	ambiguous
*	H_s	*	$D = C_h - H_s$
*	*	r	$H_s + D = C_h,\ H_s = rD$
*	*	*	$H_s + D = C_h,\ H_s = D$

\setheadfoot{⟨*headheight*⟩}{⟨*footskip*⟩}
\headheight \footskip

\setheadfoot 명령은, ⟨*headheight*⟩ 인자로 \headheight를, ⟨*footskip*⟩ 인자로 \footskip 값을 설정한다.

\setheaderspaces{⟨*headdrop*⟩}{⟨*headsep*⟩}{⟨*ratio*⟩}
\headdrop \headsep

\setheaderspaces 명령은 \setulmargins와 비슷하다. 기호 U는 앞서와 마찬가지로 상단 여백을 의미한다. H_h는 \headheight를, H_s는 \headsep을, D는 \headdrop을 뜻한다. \headdrop이란 재단된 용지의 상단과 머릿글의 상단 사이의 거리를 의미한다[3]. 이제, \setheaderspaces 매크로는 다음 관계식을 만족한다.

$$D + H_s = C_h = \text{상수}\ (= U - H_h).$$

매크로 \setheaderspaces는 페이지 상단 면주의 전후 공간을 설정한다. 이 명령에 쓰이는 인자와 그 결과는 표 [6.6]에 나와 있다. 보통 \headsep이 \headdrop보다 중요하게 취급된다.

마지막으로, 여백 문단이 있다. 여백이 문단을 놓을 정도로 충분히 주어졌는지를 보고 디자인 절차의 마지막에 가서 고려하는 것이다. 그림 (6.3)은 짝수쪽 페이지의 여백 문단을 만드는 인자들을 보여주고 있다. 몇 가지 인자들은 홀수쪽의 경우 용지의 다른 편에 놓일 것이다.

\setmarginnotes{⟨*separation*⟩}{⟨*width*⟩}{⟨*push*⟩}
\marginparsep \marginparwidth \marginparpush

[3]head drop은 그림 (6.2)에는 나와 있지 않다.

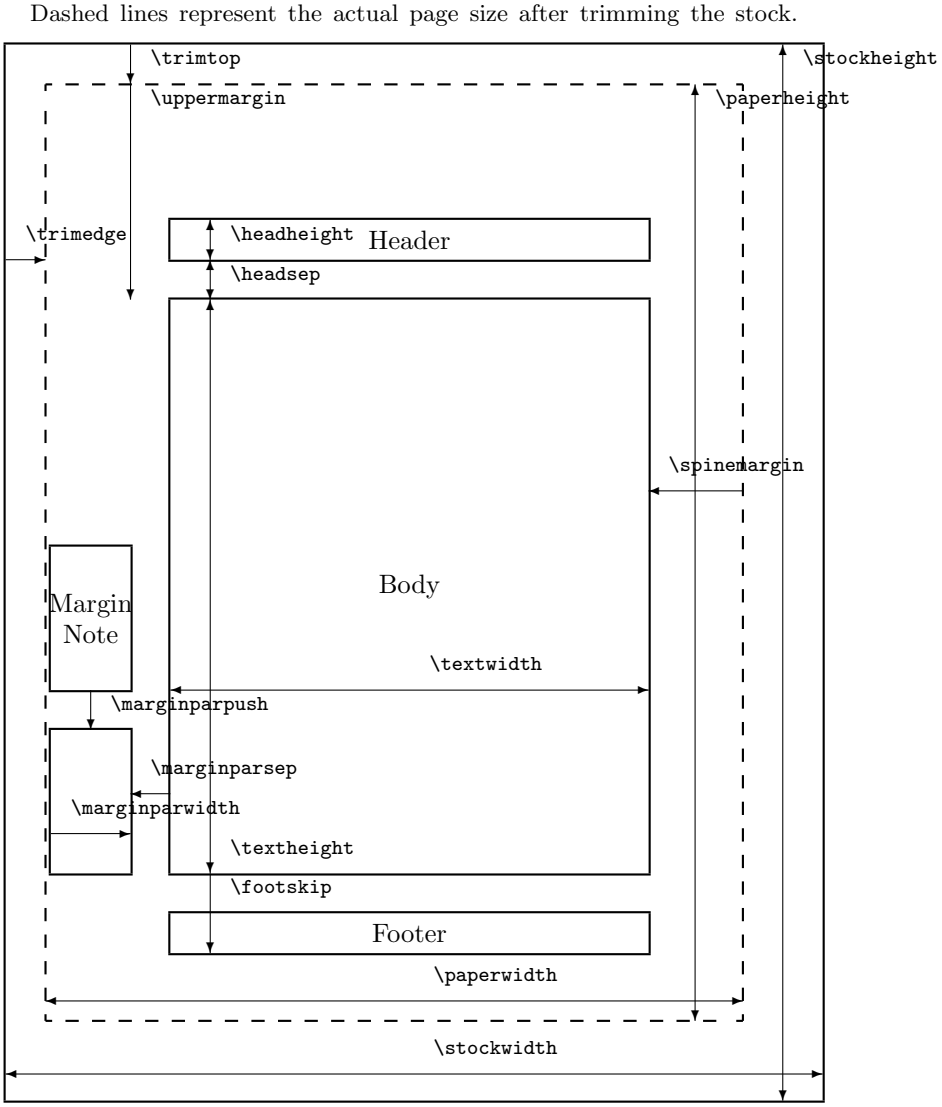


그림 6.3: memoir 클래스의 짝수쪽 페이지 레이아웃 파라미터

\setmarginnotes 명령은 여백 문단 파라미터를 설정한다. \marginparsep은 판면과 여백 문단 사이의 간격으로서 ⟨*separation*⟩ 인자로 설정한다. ⟨*width*⟩ 인자로 여백 문단 폭의 최대값인 \marginparwidth을 지정하고, 여백 문단 사이의 최소 수직 간격을 나타내는 \marginparpush 값은 ⟨*push*⟩로 설정한다.

6.6 종합

지금까지 논의된 페이지 레이아웃 인자들은 LaTeX이나 LaTeX 패키지들에서 쓰이는 것과 항상 동일한 것은 아니다. LaTeX의 파라미터들은 그림 (6.1)을 보면 나타나 있다. 페이지 레이아웃을 바꾸려면 이 클래스 명령을 사용하는 방법도 있고, \setlength나 \addtolength를 사용하여 LaTeX 파라미터 값을 직접 수정하는 방법도 있다. 후자의 방법을 택하기로 하였다면, 표준 LaTeX 파라미터에는 없는 이 클래스의 파라미터 값들은 바뀌지 않을 것이다.

사용자 자신의 페이지 레이아웃을 설정하는 일반적인 절차는 다음과 같다.

- 최종본의 판형 크기를 결정하고, 적어도 이 크기와 같거나 큰 용지 사이즈를 선택한다.
- \setstocksize 명령으로 \stockheight와 \stockwidth를 지정한 다음, \settrimmedsize 명령으로 \paperheight와 \paperwidth 값을 정한다.
- 용지에 맞추어서 판형의 위치를 설정한다. 판형 크기와 용지 크기가 동일하다면 \settrims{0pt}{0pt}를 호출한다. 크기가 다르다면 \trimtop과 \trimedge 값을 적절하게 주고 \settrims를 설정하여 페이지를 용지 상의 적당한 위치에 앉힌다.
- 판면의 크기를 결정하여 \settypeblocksize 명령으로 \textheight와 \textwidth 값을 지정한다.
- 안쪽 마진의 크기를 선택하여 \setlrmargins 명령으로 \spinemargin과 \foremargin 값을 설정한다.

  다른 방법으로, \setlrmarginsandblock 명령을 사용하여, 양쪽 양쪽 값에 상당하는 \textwidth를 직접 지정하여도 된다.
- 상단 여백을 정하고 \setulmargins 명령으로 \uppermargin과 \lowermargin 값을 설정한다.

  다른 방법으로, \setulmarginsandblock 명령을 이용하여 상하단의 여백 값에 상당하는 \textheight를 직접 지정하는 방법도 있다.

- \headheight와 \footskip 값을 정하여 \setheadfoot 명령으로 이 값들을 설정한다.
- \headskip과 \setheaderspaces 값을 정하여 이 값과 \headdrop을 설정한다.
- 여백 문단이 필요하다면 \setmarginnotes 명령으로 \marginparsep과 \marginparwidth, \marginparpush 값을 설정한다.

위의 레이아웃 파라미터 설정 순서는 편리한 대로 하면 된다. 각각의 레이아웃 명령들은 상호 독립적이다. 한번 어떤 값이 정해졌다 해도 나중에 다시 바꿀 수 있다. 마지막에 설정한 값만이 효력이 있다.

그림 (6.2)와 그림 (6.1)을 비교해보면, 이 클래스가 제공하는 레이아웃 인자와 표준 LaTeX의 것이 다르다는 것을 알 수 있을 것이다. 그림이 모든 파라미터를 모두 보여주지 않기 때문에 편의상 표 [6.7]에 두 파라미터들을 비교해놓았다.

\checkandfixthelayout

\checkandfixthelayout 매크로는 이 클래스의 모든 레이아웃 파라미터들을 체크하여 그것이 '유의미한' 값을 가지고 있는지 (예를 들면, \textwidth가 음수값은 아닌지) 점검하고 클래스 파라미터와 달라져 있는 LaTeX 레이아웃 파라미터 값을 계산한다. 어떤 방식으로든 클래스 매크로를 이용하여 레이아웃을 바꾸었다면, \checkandfixthelayout을 불러야 변경이 효과를 발생한다. 레이아웃 상태를 확인할 수 있도록 하기 위해, 최종적인 레이아웃 인자 값들을 터미널에 표시하고 로그 파일에 기록한다.

\typeoutlayout
\typeoutstandardlayout

\typeoutlayout 명령은 문서의 어느 곳에서 쓰여지든 현재의 클래스 레이아웃 파라미터 값을 터미널에 표시하고 로그 파일에 기록한다. \typeoutstandardlayout 명령은 같은 일을 하지만 표준 파라미터들의 값만을 보여준다는 점이 다르다.

pdfLaTeX을 이용하여 memoir 문서의 PDF 버전을 만드는 경우 특별한 설정이 필요하다.

\fixpdflayout

\fixpdflayout 매크로는 pdfLaTeX이 실행되면 preamble 이후에 자동으로 불려진다. 기본 설정값이 PDF 생성에 작동하도록 하는 역할을 한다.

```
\newcommand{\fixpdflayout}{\ifpdf\ifnum\pdfoutput>0\relax
  \pdfpageheight=\the\stockheight
  \pdfpagewidth=\the\stockwidth
```

표 6.7: memoir와 LaTeX의 페이지 레이아웃 파라미터

Class	LaTeX
\stockheight	
\trimtop	
\trimedge	
\stockwidth	
\paperheight	\paperheight
\paperwidth	\paperwidth
\textheight	\textheight
\textwidth	\textwidth
\columnsep	\columnsep
\columnseprule	\columnseprule
\spinemargin	
\foremargin	
	\oddsidemargin
	\evensidemargin
\uppermargin	
\headdrop	
	\topmargin
\headheight	\headheight
\headsep	\headsep
\footskip	\footskip
\lowermargin	
\marginparsep	\marginparsep
\marginparwidth	\marginparwidth
\marginparpush	\marginparpush

```
\ifdim\pdfvorigin=0pt\pdfvorigin=1in\fi
\ifdim\pdfhorigin=0pt\pdfhorigin=1in\fi
\fi\fi}
```

첫번째 설정(\pdfpage...)은 pdfLaTeX에게 프린트될 실제 용지 크기를 전달하는 일을 한다. ...origin 설정은 이 값이 0pt로 되어 있으면 pdf origin을 TeX origin에 맞추는 일을 한다. pdfLaTeX 설정 파일이나 preamble의 앞부분에서 origin 값을 따로 설정하였다면 \fixpdflayout 매크로는 이미 설정된 값을 바꾸지 않는다. (만약 origin 값으로 0을 설정하려면, 이것을 0sp로 하라. 겉보기로는 0pt와 구별되지 않는다.)

```
\fixdvipslayout
```

\fixdvipslayout 매크로는 pdfLaTeX이 실행되는 경우가 아니면 preamble이 끝나는 지점에서 자동으로 불린다. dvips 프로세서에게, 미리보기 프로그램이나 프린터가 사용할

용지 크기에 관한 정보를 제공한다.

`latex -> dvips` 루트를 거쳐서 `.ps` 포스트스크립트 파일을 생성하는 경우 landscape 문서는 `ghostview`와 같은 뷰어에서 아래위가 반전되어 나타날 수 있다. 이런 일이 벌어진다면, preamble에 다음을 넣어두도록 하라.

```
\addtodef{\fixdvipslayout}{}{%
  \special{!TeXDict begin /landplus90{true}store end }}
```

필요하다면, `\addtodef` 명령을 반복 적용하여 `\fixdvipslayout`에 추가 코드를 덧붙일 수 있다. 예를 들면, 포스트스크립트 인쇄 special 명령을 다음과 같이 추가하는 것이 가능하다(HP 5SiMx LajerJet duplex printer의 경우에는 적용이 된다).

```
\special{!TeXDict begin <</Duplex true>> setpagedevice end} % duplex
\special{!TeXDict begin <</Tumble true>> setpagedevice end} % short side binding
```

예제

가령, A4와 US letterpaper에 모두 적당하도록 판형을 만들어서 최소한으로만 재단하게 하고 싶은 경우를 생각해보자. 판면의 폭은 행당 문자수가 최적이 되도록 해야 하고 판면의 가로 길이와 세로 길이는 황금비를 유지하도록 하고 싶다. 이럴 경우 텍스트는 판형 페이지의 상단에서 50pt 떨어져서 시작해야 한다. `\headheight`와 `\footskip`은 기본값으로 하도록 하자. 바깥쪽 여백과 안쪽 여백의 비율, 그리고 상단 면주의 위쪽 공간과 아래쪽 공간은 황금비가 되도록 한다. 여백 문단에 대해서는 고려하지 않기로 한다.

직접 계산할 수도 있지만 LaTeX으로 하여금 계산하도록 할 수도 있다. 여기서는 계산을 LaTeX에게 맡기자. 먼저, A4(297×210mm)와 letterpaper(11×8.5in)에서 잘라낼 수 있는 가장 큰 크기를 찾아본다. 이것은 가로는 A4, 세로는 letterpaper에서 취할 수 있다(A4는 letterpaper에 비해서 가로 길이는 짧고 세로 길이는 길다).

```
\settrimmedsize{11in}{210mm}{*}
```

용지 크기는 클래스 옵션으로 지정한다. 그러나 우리는 용지를 판형 크기로 재단할 수 있도록 해야 한다. 쉽게 가는 방법은 용지의 바깥쪽과 아래쪽만을 재단하는 것이다. 그러면 `\trimtop`은 0이 된다. `\trimtop`과 `\trimedge`는 다음과 같이 쉽게 지정할 수 있다.

```
\setlength{\trimtop}{0pt}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

판면 크기를 설정하는 것도 쉽다.

```
\settypeblocksize{*}{\lxvchars}{1.618}
```

그리고 상하 여백을 다음과 같이 설정한다.

```
\setulmargins{50pt}{*}{*}
```

안쪽과 바깥쪽 여백은 황금비 값에 의하여 다음과 같이 설정한다.

```
\setlrmargins{*}{*}{1.618}
```

이제 남은 문제는 \headdrop과 \headsep을 지정하는 것이다. 이것도 비율을 이용하면 간단하다.

```
\setheaderspaces{*}{*}{1.618}
```

마지막으로 이상의 레이아웃 설정을 확인한다.

```
\checkandfixthelayout
```

이 사용안내서의 레이아웃

이 매뉴얼의 페이지 레이아웃은 preamble에 정의되어 있다. 코드는 다음과 같다.

```
\settrimmedsize{11in}{210mm}{*}
\setlength{\trimtop}{0pt}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settypeblocksize{7.75in}{33pc}{*}
\setulmargins{4cm}{*}{*}
\setlrmargins{1.25in}{*}{*}
\setmarginnotes{17pt}{51pt}{\onelineskip}
\setheadfoot{\onelineskip}{2\onelineskip}
\setheaderspaces{*}{2\onelineskip}{*}
\checkandfixthelayout
```

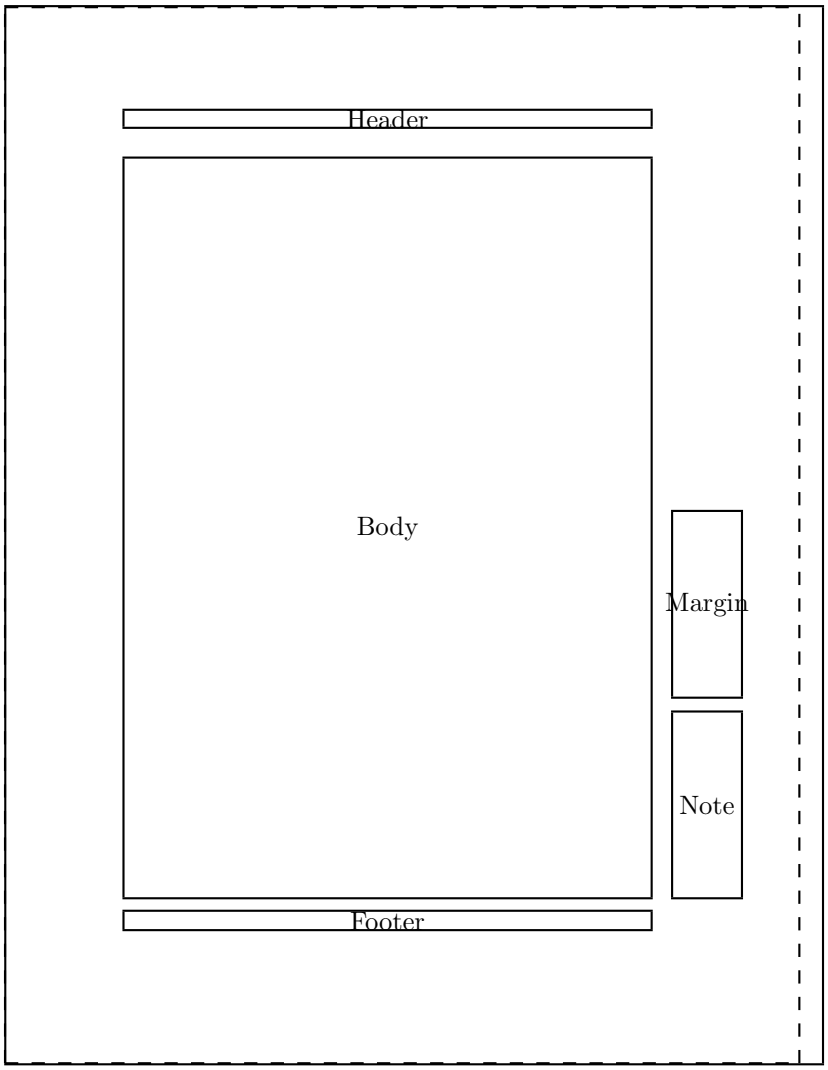
레이아웃 모양은 그림 (6.4)에 보였다. 이 그림에는 위의 코드로 만들어지는 파라미터 결과값도 근사치로 열거되어 있다.

처음에 나는 레이아웃을 §6.6에 정의된 것과 같이 설정했었다. 내게는 그것이 합리적으로 보였기 때문이다. 그러나 나중에 위와 같은 설정을 사용하기로 했는데, 그것은 이 매뉴얼을 인쇄하는 경우 종이를 절약하도록 하기 위해서였다. 판면을 넓게 잡으면, LaTeX 소스 코드처럼 하이픈 처리되지 않는 텍스트를 TeX이 처리하기가 더 쉬워진다.

Andreas Mathias는 Rolf Niepraschk[4]를 통해서, A4 용지에 매뉴얼을 인쇄하려면 다음과 같은 설정이 좋을 것이라고 제안하였다.

[4] 2002/02/05일자 niepraschk@ptb.de로부터 온 Email

Dashed lines represent the actual page size after trimming the stock.



Lengths are to the nearest pt.

\stockheight = 795pt	\stockwidth = 614pt
\pageheight = 795pt	\pagewidth = 598pt
\textheight = 556pt	\textwidth = 396pt
\trimtop = 0pt	\trimedge = 17pt
\uppermargin = 114pt	\spinemargin = 90pt
\headheight = 12pt	\headsep = 24pt
\footskip = 24pt	\marginparsep = 17pt
\marginparpush = 12pt	\columnsep = 10pt
\columnseprule = 0.0pt	

그림 6.4: 이 문서의 홀수쪽 레이아웃

```
\documentclass[a4paper]{memoir}
...
\settrimmedsize{297mm}{210mm}{*}
\setlength{\trimtop}{0pt}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settypeblocksize{634pt}{448.13pt}{*}
\setulmargins{4cm}{*}{*}
\setlrmargins{*}{*}{1.5}
\setmarginnotes{17pt}{51pt}{\onelineskip}
\setheadfoot{\onelineskip}{2\onelineskip}
\setheaderspaces{*}{2\onelineskip}{*}
\checkandfixthelayout
```

제 7 장

타이틀(Title)과 요약문(abstract)

7.1 들어가는 말

\maketitle 명령의 조판 형태는 LaTeX 표준 클래스에서는 바꿀 수 없는 것으로 보아야 한다. 그러나 이 클래스에서는 타이틀 정보, 즉 \title, \author, \date 명령의 형태를 수정할 수 있도록 일련의 형식화 명령을 제공한다. 그리고 이 명령의 값을 유지해서 나중에 문서에서 다시 활용할 수 있도록 한다. 이 클래스는 \maketitle 명령이 사용된 후에는 타이틀 만들기에 사용된 명령 값이 자동으로 지워지는 기능을 방지하고 있다. 그러므로 하나의 문서 안에서 동일한 타이틀이 여러 번 나오게 할 수도 있다. 예를 들면 반표지와 전표지를 별도로 만들면서 같은 타이틀 정보를 식자할 수 있는 것이다. 또한, \maketitle 명령이 사용하는 타이틀 구성요소에 새로운 것을 추가로 정의할 수도 있다. \thanks 명령의 기능이 확장되어, thanks 주석의 표지 부호와 레이아웃을 다양하게 설정할 수 있게 하였다.

twocolumn 문서에서 onecolumn abstract를 어떻게 구현하는가에 대한 질문이 뉴스그룹(comp.text.tex)에 자주 보인다. ctt의 해결책들이 FAQ로서 제시되고 있는데, 이 클래스는 이 문제에 대한 좀더 저자 친화적인 방식의 해결책을 제공한다. 나아가, 일반적인 abstract 환경의 조판을 위한 제어 기능을 추가적으로 제공한다.

7.2 표지(titles)

타이틀 스타일 설정

이 부분은 titling 패키지를 재구현한 것이다.

이 클래스는 설정 가능한 \maketitle 명령을 제공한다. \maketitle 명령의 정의는 본질적으로 다음과 같다.

```
\newcommand{\maketitle}{%
   \vspace*{\droptitle}
   \maketitlehooka
   {\pretitle \title \posttitle}
   \maketitlehookb
   {\preauthor \author \postauthor}
   \maketitlehookc
   {\predate \date \postdate}
   \maketitlehookd
   \thispagestyle{title}
}
```

여기서 *title* 페이지스타일은 처음에는 *plain* 페이지 스타일과 동일하게 설정된다. \maketitle 내부의 여러 매크로들은 아래 설명하는 바와 같다.

\pretitle{⟨*text*⟩} \posttitle{⟨*text*⟩}
\preauthor{⟨*text*⟩} \postauthor{⟨*text*⟩}
\predate{⟨*text*⟩} \postdate{⟨*text*⟩}

이 여섯 개의 명령은 각기 한 개의 인자를 갖는다. 인자로 오는 ⟨*text*⟩는 \maketitle의 표준적 구성요소의 식자를 통제한다. \title은 \pretitle과 \posttitle 사이에서 처리된다. 즉, 다음과 같다.

```
{\pretitle \title \posttitle}
```

마찬가지로 \author와 \date 명령에도 이 방식이 적용된다. 이 명령들은 처음에는 report 클래스의 \maketitle 명령의 일반적 결과를 흉내내도록 설정되어 있다. 이 명령들의 기본값 정의는 다음과 같다.

```
\pretitle{\begin{center}\LARGE}
\posttitle{\par\end{center}\vskip 0.5em}
\preauthor{\begin{center}
           \large \lineskip 0.5em%
           \begin{tabular}[t]{c}}
\postauthor{\end{tabular}\par\end{center}}
\predate{\begin{center}\large}
\postdate{\par\end{center}}
```

이 값들을 변경하면 서로 다른 효과가 나타난다. 예컨대 타이틀은 오른쪽 정렬된 sans-serif로 하고 날짜는 왼쪽 정렬된 small caps로 하려면 다음과 같이 하라.

```
\pretitle{\begin{flushright}\LARGE\sffamily}
\posttitle{\par\end{flushright}\vskip 0.5em}
\predate{\begin{flushleft}\large\scshape}
\postdate{\par\end{flushleft}}
```

\droptitle

\maketitle 명령은 타이틀을 페이지의 특정 세로 위치에 놓는다. \droptitle 길이를 변경시킴으로써 타이틀이 놓일 세로 위치를 바꿀 수 있다. 이 값이 양수이면 타이틀은 아래로 이동하고 음수이면 위로 옮겨진다. 기본값은 다음과 같이 정의되어 있다.

```
\setlength{\droptitle}{0pt}
```

\maketitlehooka \maketitlehookb \maketitlehookc \maketitlehookd

이 네 개의 후킹 명령은 \maketitle에 새로운 요소를 추가하기 위한 것이다. 처음에는 아무것도 정의되어 있지 않으므로 각 명령을 재정의한다. 예를 들면, 몇몇 출판사들은 논문이 인쇄된 장소에 대한 언급을 제목줄 직전에 삽입하고 싶어한다. 아래 보인 것과 같이, \published 명령을 정의하여 출판관련 정보를 넣어두면 \maketitle 명령이 불릴 때 자동으로 그 내용을 인쇄해주게 된다.

```
\newcommand{\published}[1]{%
   \gdef\puB{#1}}
\newcommand{\puB}{}
\renewcommand{\maketitlehooka}{%
   \par\noindent \puB}
```

다음과 같이 하면 출판 사항과 보통의 타이틀 정보를 함께 프린트할 수 있다.

```
\published{Originally published in
          \textit{The Journal of ...}\thanks{Reprinted with permission}}
...
\maketitle
```

\thanks 명령을 아무런 추가적인 조치 없이 \published 명령의 인자 안에서 사용하였음에 주의하라.

\begin{titlingpage} text \end{titlingpage}

표준 클래스에는 titlepage 옵션이 있는데, 이것은 \maketitle이 불리면 번호붙지 않은 페이지를 만들고 타이틀 구성요소를 놓은 다음 새로운 페이지를 페이지 번호 1로 시작하게 만들어준다. 또, 표준 클래스에는 titlepage 환경이 있는데, 이 환경은 시작하면서 번호붙지 않은 새로운 페이지로 시작하고, 종료되면 새로운 페이지를 시작하면서 페이지 번호를 1로 붙여준다. 이 타이틀 페이지 안에 무엇을 둘 것이며 어디에 놓을 것인가는 전적으로 사용자 책임이다. \maketitle이 이것은 titlepage 환경 안에서 사용되면 또다시 새로운 페이지가 시작한다.

이 클래스에서는 titlepage 옵션이나 `titlepage` 환경 어느 것도 제공하지 않는다. 그 대신 `titlingpage` 환경이 있는데 이것은 titlepage 옵션과 `titlepage` 환경의 중간쯤 된다. `titlingpage` 환경 안에는 `\maketitle`을 쓸 수 있고 그밖의 것도 무엇이든 올 수 있다. 여기서는 *titlingpage* 페이지스타일이 사용되며 환경이 종료되면 그 다음 정상적인 페이지의 번호를 1로 만든다. *titlingpage* 페이지스타일은 애초 *empty* 페이지스타일과 동일하게 설정되어 있다.

예를 들면, 타이틀과 abstract를 같은 타이틀 페이지에 두고 페이지스타일을 *plain*으로 하려면 다음과 같이 한다.

```
\begin{document}
\begin{titlingpage}
\aliaspagestyle{titlingpage}{plain}
\setlength{\droptitle}{30pt} lower the title
\maketitle
\begin{abstract}...\end{abstract}
\end{titlingpage}
```

타이틀 정보는 편집 영역 폭을 기준으로 가운데 정렬되는 것이 기본값이다. 이따금 `comp.text.tex` 뉴스그룹에는 타이틀 페이지의 타이틀 정보를 편집 영역 기준이 아니라 실제 페이지를 기준으로 중앙 정렬하는 방법이 무엇이냐고 묻는 질문이 올라온다. 편집 영역이 실제 페이지에 대해서 가운데 오도록 디자인되어 있다면 centering만으로 충분할 것이다. 만약 편집 영역이 실제 페이지에 대해서 가운데가 아니라면 타이틀 정보는 편집 영역이 중앙을 기준으로 이동해 있는 만큼 수평 이동을 시켜주어야 할 것이다. 가장 쉬운 해결책은 `\calccentring` 명령과 `adjustwidth*` 환경을 쓰는 것이다. 예를 들면 다음과 같다.

```
\begin{titlingpage}
  \calccentering{\unitlength}
  \begin{adjustwidth*}{\unitlength}{-\unitlength}
    \maketitle
  \end{adjustwidth*}
\end{titlingpage}
```

`\title{`⟨*text*⟩`}` `\thetitle`
`\author{`⟨*text*⟩`}` `\theauthor`
`\date{`⟨*text*⟩`}` `\thedate`

일반적인 문서 클래스에서 (⟨*text*⟩)의 위치에 오는 `\title`, `\author`, `\date` 매크로의 내용은 `\maketitle` 명령에 단 한 번 쓰인다. 그러나 이 클래스는 `\thetitle`, `\theauthor`, `\thedate` 명령을 통해서 이 타이틀 요소들을 필요하다면 문서의 뒷부분에서 다시 쓸 수 있도록 보존해 준다.

\and \andnext

\and 매크로는 \author 명령의 인자 안에서 쓰여서 저자들의 이름 사이에 적당한 추가 공백을 넣어준다. 이 클래스에서 제공하는 \andnext 매크로는 공백이 아니라 새로운 줄을 넣어준다. \theauthor 매크로는 \and와 \andnext를 쉼표로 바꾸어서 보존한다.

이 클래스는 표준 클래스가 \maketitle이 사용되고 나면 타이틀 관련 명령의 내용을 자동적으로 삭제하는 방식을 따르지 않는다. 원한다면 \title, \author, \date, \maketitle 명령을 여러 번 쓸 수 있다. 예컨대, 어떤 보고서에 맨 처음에 타이틀 페이지가 있고 그 다음에 요약문이 첨부되고, 그 뒤에 본문이 시작될 때 조금 다르게 표현된 제목을 한 번 더 둔다고 가정하자. 다음과 같이 처리하면 된다.

```
\title{Cover title}
...
\begin{titlingpage}
\maketitle
\end{titlingpage}
...
\title{Body title}
\maketitle
...
```

\killtitle \keepthetitle \emptythanks

\killtitle 매크로는 \thetitle 등 타이틀 관련 명령을 전부 사용하지 못하도록 만든다. 이 명령을 사용하면 \thetitle 등의 명령이 더이상 필요하지 않을 경우에 매크로 저장 공간을 절약할 수 있다. 이 클래스 매뉴얼 버전이 이 명령을 사용하여 표준 클래스의 자동 삭제와 동일한 방식으로 처리하고 있다. \keepthetitle 명령은 이와 유사한 명령이지만 \thetitle, \theauthor, \thedate 명령을 보존하고 그 이외에는 모두 삭제한다.

\emptythanks는 앞서 \thanks에서 사용된 모든 텍스트를 지운다. 이 명령은 \maketitle이 여러 번 쓰일 때 유용하다. 즉, \thanks 명령이 보존되어 있으면 그 텍스트가 매번 나타날 것이므로 이것을 피하려면 \emptythanks 명령을 맨처음 사용된 \maketitle 이후 \maketitle 앞에서 지시해주면 된다.

thanks 스타일 설정

이 클래스는 \thanks 명령을 설정할 수 있다.

\thanksmarkseries{⟨*format*⟩} \symbolthanksmark

\thanks{}는 타이틀이나 각주 양쪽에 심볼 표지가 붙는다. \thanksmarkseries 명령으로 표지 스타일을 바꿀 수 있다. ⟨*format*⟩ 인자는 숫자 표기 형식 명칭 중의 하나를 쓴다. 형식은 카운터 형식과 동일하지만 백슬래시 없이 지정해야 한다. \thanks 표지를 심볼문자가 아닌 알파벳 소문자로 표시하고 싶다면 다음과 같이 한다.

```
\thanksmarkseries{alph}
```

편의를 위해서 \symbolthanksmark 명령이 마련되어 있다. 이 명령이 지시되면 각주 심볼이 사용된다. ⟨*format*⟩에 쓰일 수 있는 지시자는, arabic, roman, Roman, alph, Alph, fnsymbol 들이다.

\continuousmarks

\thanks 명령은 footnote 카운터를 이용한다. 그리고 타이틀 만들기가 끝나면 각주 번호가 1에서 시작할 수 있도록 카운터는 0으로 되돌려진다. 만약 카운터가 0이 되지 않도록 하고 싶다면 그저 \continuousmarks라고 지시하라. thanks 표지로 숫자를 사용하였다면 이렇게 해야 할 것이다.

\thanksheadextra{⟨*pre*⟩}{⟨*post*⟩}

\thanksheadextra 명령은 ⟨*pre*⟩와 ⟨*post*⟩를 타이틀 문단에서 thanks 표지의 전후에 넣는다. 기본값은 아무 것도 넣지 않는 것이다. 예를 들면, 타이틀 표지를 괄호로 둘러싸고 싶을 때는 다음과 같이 한다.

```
\thanksheadextra{(}{)}
```

\thanksmark{⟨*n*⟩}

이따금 동일한 thanks 텍스트를 적용하고 싶을 때가 있다. 예컨대, 여섯 명의 저자 가운데 처음 세 명에 대해서 동일한 thanks를 붙이고 나머지는 하나씩 붙여서 네 개의 thanks만을 적용하는 경우가 그러한 예이다. \thanksmark{⟨*n*⟩} 명령은 \footnotemark[⟨*n*⟩]과 비슷하게, ⟨*n*⟩번째 \thanks 명령에 독립적인 thanks 표지를 붙이면서 각주 위치의 thanks에는 변경을 가하지 않는다. 다음 보기에서 Alpha와 Omega 저자가 동일한 표지를 갖게 된다.

```
\title{The work\thanks{Draft}}
\author{Alpha\thanks{ABC},
        Beta\thanks{XYZ} and
        Omega\thanksmark{2}}
\maketitle
```

\thanksmarkstyle{⟨*defn*⟩}

기본값은 바닥글의 thanks 표지가 상첨자(superscript)로 식자되는 것이다. 이 클래스에서 다음과 같이 정의되어 있다.

```
\thanksmarkstyle{\textsuperscript{#1}}
```

여기서 #1은 thanks 표지 기호로 바뀌게 될 부분이다. 표지기호 스타일은 원한다면 바꿀 수 있다. 예를 들어 표지기호를 괄호로 둘러싸고 베이스라인에 normal size로 식자하게 하려면 다음과 같이 한다.

```
\thanksmarkstyle{(#1)}
```

\thanksmarkwidth

각주 위치의 thanks 표지는 \thanksmarkwidth 폭을 가진 박스 안에 오른쪽 정렬된다. thanks 텍스트는 이 박스 이후에 시작한다. 애초값은

```
\setlength{\thanksmarkwidth}{1.8em}
```

로 설정되어 있다.

\thanksmarksep

\thanksmarksep이라는 길이는 thanks 텍스트의 다음 줄 이후가 표지 박스를 기준으로 어느 정도나 들여밀기 되는지를 결정한다. 예를 들어 다음과 같이 하면,

```
\setlength{\thanksmarksep}{0em}
```

이어지는 줄은 thanks 텍스트의 왼쪽 끝을 기준으로 정렬될 것이다. 반면,

```
\setlength{\thanksmarksep}{-\thanksmarkwidth}
```

이렇게 하면 thanks 텍스트의 두번째 이후 줄은 왼쪽 정렬될 것이다. 이것이 기본값이다.

\thanksfootmark

각주 위치의 thanks 표지 영역은 \thanksfootmark로 식자된다. 코드는 대체로 다음과 같다.

```
\newcommand{\thanksfootmark}{%
  \hbox to\thanksmarkwidth{\hfil\normalfont%
     \thanksscript{\thanksfootpre \thefootnote \thanksfootpost}}}
```

\thanksfootpre와 \thanksfootpost는 \thanksfootextra 매크로에 의하여 설정된다. \thanksfootmark 정의를 변경할 필요는 없다. 그러나 이 매크로고 사용하고 있는 매크로들 가운데 하나나 둘 정도는 변경하고 싶어질 수도 있다.

\makethanksmark
\makethanksmarkhook

\makethanksmark 매크로는 thanks 표지(\thanksfootmark)와 thanks 텍스트를 모두 식자한다. 이 기본 정의를 바꿀 필요는 없다. 다만 \makethanksmark가 thanks를 식자하기 전에 부르는 \makethanksmarkhook를 바꾸어야 할 때는 있을 수 있다. 기본 정의는 다음과 같이,

```
\newcommand{\makethanksmarkhook}{}
```

아무것도 하지 않는 것이 기본값이다.

\makethanksmarkhook를 재정의하여 뭔가 쓸만한 일을 시켜보자. 예를 들면 윗쪽 각주와 조금 간격을 두고 싶을 때라면,

```
\renewcommand{\makethanksmarkhook}{\fontsize{8}{11}\selectfont}
```

이렇게 재정의해볼 수 있다. 여기서 8과 11은 폰트의 크기와 baselineskip 값을 각각 나타낸다. 이 예에서 8pt는 일반적인 10pt 문서에서 \footnotesize이고, 11pt는 11pt 문서에서 \footnotesize에 해당하는 baselineskip 값이다(10pt 문서에서 이 값은 9.5pt이다). 이 값들을 적절하게 조절해보라.

\thanksrule
\usethanksrule
\cancelthanksrule

\thanks 텍스트는 titlingpage 환경 안에서 선 없이 나타난다. 만약 선을 긋고 싶으면 \usethanksrule 매크로를 \maketitle 앞에 선언한다. \thanksrule에 설정된 모양으로 선을 그어준다. \thanksrule의 애초 정의는 \footnoterule과 같고, 이것은 preamble이 끝나는 지점에서 정의된다. \thanksrule의 정의를 바꾸려 한다면 \begin{document} 이후에 해야 한다. \thanksrule이 재정의되고 \usethanksrule 명령이 지시되면, 재정의된 rule이 각주에도 쓰이게 된다. \cancelthanksrule 명령은 그 이후부터 기본적인 \footnoterule을 사용하도록 한다. 원한다면 그 후에 \usethanksrule을 다시 쓸 수도 있다.

각주와 thanks 주석의 수직 위치, 그리고 \footnoterule을 결정하는 파라미터는 동일하다. 216 쪽의 그림 (13.1)을 보라. 이 가운데 일부를 바꾸면 각주의 수직 간격과 thanks 노트가 서로 다르게 만들 수 있다.

7.3 요약문(Abstracts)

이 클래스의 이 부분은 대부분 abstract 패키지를 재구현한 것이다.

report와 article 클래스의 abstract 조판 형식은[1] 클래스 옵션에 좌우된다. 그것은 다음과 같다.

- titlepage class option: abstract 제목줄은 굵은 폰트로 가운데 정렬 식자된다. 텍스트는 normal font와 normal width를 갖는다.
- twocolumn class option: abstract 제목줄은 번호붙지 않은 절(section)로 식자된다. 텍스트는 normal font와 normal width의 single column이다.
- Default (위의 어떤 옵션도 주지 않았을 때) : abstract 제목줄은 small bold font로 가운데 정렬된다. 텍스트는 small font이고 quotation 환경과 같은 들여밀기 문단이 된다.

이 클래스는 abstract 환경을 제공하고, abstract의 조판을 수정할 수 있도록 하고 있다.

```
\begin{abstract} text \end{abstract}
```

abstract 환경에 특별할 것은 아무것도 없다. 형식은 아래 설명하는 매크로에 의하여 제어된다.

twocolumn 문서에서 onecolumn abstract를 만드는 자주 묻는 질문에 대한 답은 다음과 같은 것이다.

```
\documentclass[twocolumn...]{...}
...
\twocolumn[
   \begin{@twocolumnfalse}
     \maketitle                need full-width title
     \begin{abstract}
        abstract text...
     \end{abstract}
   \end{@twocolumnfalse}
]
... hand make footnotes for any \thanks commands
...
```

```
\begin{onecolabstract} text \end{onecolabstract}
\saythanks
```

[1] book 클래스에는 abstract 환경이 없다.

이 클래스는 onecolabstract 환경을 제공한다. 이를 이용하면 twocolumn 문서에서 onecolumn abstract를 식자할 수 있다.

```
\documentclass[twocolumn...]{memoir}
...
\twocolumn[
   \maketitle               need full-width title
   \begin{onecolabstract}
     abstract text...
   \end{onecolabstract}
]
\saythanks  typesets any \thanks commands
...
```

\saythanks 명령은 \thanks 텍스트가 일반적인 모양으로 식자되도록 해준다.

원한다면 onecolabstract 환경을 abstract 환경의 위치에 둘 수도 있다. 사실 반드시 \twocolumn의 옵션 인자로 쓰일 필요는 없다. onecolabstract는 내부적으로 abstract를 사용한다.

```
\abstractcol
\abstractintoc
\abstractnum
\abstractrunin
```

abstract의 일반적인 형식은 중앙정렬된 굵은 타이틀에 텍스트는 small font로 하고 문단의 좌우 여백을 조금 안쪽으로 들이는 것이다.

\abstractcol 선언은 abstract가 twocolumn 클래스 옵션 문서에서 보통 문단에 번호 붙지 않은 chapter로 식자되도록 한다. \abstractintoc 선언은 abstract 타이틀을 ToC에 추가하도록 한다. \abstractnum 선언은 abstract가 번호붙은 chapter로 식자되도록 지정하고, \abstractrunin은 abstract 타이틀이 문단 첫머리로 나타나도록 한다. 이 두 선언은 상호 배타적이다. \abstractnum 선언은 abstract가 \frontmatter에 있을 때는 효력이 없음에 주의하라.

```
\abstractnamefont
\abstracttextfont
```

이 두 명령을 재정의하면 폰트를 바꿀 수 있다. 각각 abstract 이름(\abstractname으로 지정되어 있음)과 abstract 텍스트의 글꼴을 재설정하는 데 이용된다. 기본 설정은 다음과 같다.

```
\newcommand{\abstractnamefont}{\normalfont\small\bfseries}
\newcommand{\abstracttextfont}{\normalfont\small}
```

```
\absleftindent \absrightindent
\absparindent \absparsep
```

abstract의 현재 버전은 list 환경을 이용하여 조판한다. 위의 네 가지는 \setlength나 \addtolength로 조절할 수 있는 길이변수로서, 왼쪽 및 오른쪽 마진과 문단의 들여밀기, 문단 사이의 수직 벌림값을 바꿀 수 있도록 하는 것이다. 기본값은 twocolumn 옵션이 주어졌느냐 그렇지 않으냐에 따라 다르다.

```
\abslabeldelim{⟨text⟩}
```

\abstractrunin 선언이 주어지면 문단 첫머리에 abstract 이름이 식자된다. 즉 이것은 abstract 텍스트의 첫 줄 첫번째 텍스트 일부인 것처럼 나타나는 것이다. \abslabeldelim의 ⟨*text*⟩ 인자는 heading 직후에 식자된다. 기본값으로는 아무것도 지정되어 있지 않지만 예를 들어 \abstractname 다음에 콜론과 약간의 공백을 지정하고 싶다면, 다음과 같이 한다.

```
\abslabeldelim{:\quad}
```

```
\absnamepos
```

\abstractrunin 선언이 사용되지 않은 경우에 abstract heading은 자체 환경으로 식자된다. 이것은 \absnamepos로 지정되는 것으로, 기본 정의는 다음과 같다.

```
\newcommand{\absnamepos}{center}
```

flushleft, center, flushright 가운데 하나를 지정할 수 있는데, 각각 heading을 왼쪽, 중앙, 오른쪽으로 정렬한다. 특정한 패키지를 사용하였다면 그 패키지가 지원하는 별도의 환경을 지정하여도 된다.

```
\abstitleskip
```

\abstractrunin 선언을 사용할 때, \abstitleskip은 헤딩 앞에 약간의 공백을 둔다. 예를 들어 \absparindent가 0이 아니라면

```
\setlength{\abstitleskip}{-\absparindent}
```

와 같이 하면 헤딩이 왼쪽 끝에서 시작할 것이다.

\abstractrunin이 아니라면 \abstitleskip은 abstract 이름과 abstract 텍스트 사이에 수직 간격을 추가로 넣어준다. 이 값은 음수값일 수도 있다.

8 문서의 장절구분

이 장은 이 장 맨 마지막에 나오는 절 헤딩 스타일 설정으로 조판되었다.

8.1 들어가는 말

이 장에서는 먼저 하나의 책이 여러 장절 구분으로 나누어지는 문제를 검토한 다음, 그것을 조판하는 명령에 대해서 다루려고 한다.

그 다음에 장절 헤딩의 모양을 수정하는 방법에 대해서 기술하겠다. 여기서 기술된 기능들은 titlesec과 sectsty 패키지를 이용하면 거의 비슷하게 얻을 수 있는 것이지만 사용되는 명령은 다르다.

8.2 책의 구성부분

앞서 말한 바대로, 한 권의 책은 주요한 세 부분으로 논리적으로 나누어진다. front, main, back matter들이 그것이다. 여기에 대응하는 LaTeX 명령이 각각 `\frontmatter`, `\mainmatter`, `\backmatter`이다.

`\frontmatter` `\frontmatter*`

`\frontmatter` 선언은 페이지 번호가 소문자 로마 숫자로 인쇄되고 페이지 번호매김이 i부터 시작하며, 장절명령에 숫자를 붙이지 않는다. 캡션, 수식 등은 일련번호가 붙는다. 별표붙은 명령 `\frontmatter*`는 페이지 번호의 인쇄 형식이나 페이지 번호붙임이 바뀌지 않는다는 점만을 제외하고 동일하다.

`\mainmatter` `\mainmatter*`

`\mainmatter` 선언은 문서가 시작할 때의 기본값인데, 페이지 번호는 아라비아 숫자로 인쇄되고 페이지 번호매김은 1부터 시작한다. 장절명령에는 번호가 붙는다. 떠다니는 개체의 캡션, 수식 등은 chapter 기준으로 번호가 붙는다. 별표붙은 명령 `\mainmatter*`는 이와 동일하지만 페이지 번호매김이나 페이지 번호의 인쇄 형식을 바꾸지 않는다는 점이 다르다.

```
\backmatter
```

`\backmatter` 선언은 페이지 번호붙임이나 난외표제의 형식에는 아무런 변화를 주지 않는다. 그러나 장절표제에는 번호가 붙지 않으며, 캡션 등의 번호는 이어진다.

8.3 장절구분 명령

memoir 클래스는 문서를 일곱 단계의 장절 단위로 구분한다. 범위는 편(part), 장(chapter)에서 sub-paragraph까지이다. 각각 `\part`, `\chapter`, `\section`, `\subsection` 명령을 사용하여 장절을 구분한다. 만약 이보다 더 하위 단위 구분이 필요하다면 `\subsubsection`, `\paragraph`, `\subparagraph`까지 쓸 수 있다. chapter를 제외하고 나머지 모든 장절 구분 명령은 동일한 형식을 가지고 있다. 따라서 하나하나 설명하기보다 `\section` 명령과 다른 것을 함께 다루고자 한다.

```
\section[⟨toc-title⟩][⟨head-title⟩]{⟨title⟩}
\section*{⟨title⟩}
```

이 명령에는 두 가지 형식이 있다. 별표붙은 명령이 더 간단하므로 그것에 대해 먼저 지적하겠다. 별표붙은 명령을 쓰면 ⟨*title*⟩을 주어진 장절구분 단위에 해당하는 형식으로 문서에 식자한다. 별표붙지 않은 명령 형식도 ⟨*title*⟩을 문서에 식자하는 것은 같지만 번호가 붙는다. 차례(ToC)와 난외 면주에 넣을 장절 타이틀을 다음과 같이 별도로 지정할 수 있다.

- 옵션 인자 없이 사용: ⟨*title*⟩이 장절명령과 면주 및 목차에 사용된다.
- 한 개의 옵션 인자: ⟨*title*⟩이 장절명령 타이틀로 이용되고 ⟨*toc-title*⟩은 목차와 면주 타이틀로 사용된다.
- 두 개의 옵션 인자: ⟨*title*⟩은 장절명령 타이틀로만 사용된다. ⟨*toc-title*⟩은 목차 표제로 이용되고 ⟨*head-title*⟩은 면주 타이틀로 쓰인다.

`\section` 명령은 `\subsection` 등의 번호를 재설정한다. 대부분의 장절명령은 ⟨*title*⟩이 명령이 실행된 위치에 놓인다. 다만 `\part`와 `\chapter` 명령은 약간 다르게 동작한다.

`\part`⟨*title*⟩ 명령은 part name (기본값은 `Part`), 번호와 ⟨*title*⟩을 새로운 페이지로 만든다. part의 번호붙이기는 chapter의 번호붙이기에 영향을 미치지 않는다.

`Part` 등 LaTeX의 장절명령의 이름 기본값은 나중에 보이겠다.

【memhangul】 한글 문서에서 part와 chapter의 번호붙이기는 특별한 형식을 가진다. 즉, 영문문서에서 Part I 처럼 붙는 편명을 "제 I 편"과 같이 붙여야 하는 것이다. 이 문제는 memhangul-ucs 패키지에 의하여 해결되지만, memoir 클래스에는 없는 추가적인 명령을 필요로 한다. ■

```
\chapter[⟨toc-title⟩][⟨head-title⟩]{⟨title⟩}
\chapter*[⟨head-title⟩]{⟨title⟩}
```

\chapter 명령은 새로운 페이지를 시작하고 장이름(기본값은 Chapter), 번호, ⟨*title*⟩을 판면의 윗쪽에 둔다. section 등의 번호는 재설정된다. 옵션 인자가 주어지지 않으면 ⟨*title*⟩이 ToC 엔트리와 면주에 사용된다. 옵션 인자가 하나 주어지면 ⟨*toc-title*⟩은 ToC 엔트리와 면주에 사용된다. 두 개의 옵션 인자가 주어지면 ⟨*head-title*⟩이 면주에 쓰인다.

별표붙은 \chapter* 명령은 새로운 페이지를 시작하고 ⟨*title*⟩을 페이지의 윗쪽에 둔다. 이 명령은 ToC 엔트리를 만들지 않고 번호와 면주를 수정하지도 않는다. 옵션 인자 ⟨*head-title*⟩이 주어지면 면주에 나타난다. 옵션 인자를 사용하게 되면 secnumdepth 카운터가 maxsecnumdepth (아래 설명을 보라) 으로 설정되는 효과가 생긴다.

그런데 article 문서 옵션이 주어지면 좀 다른 일이 벌어진다. 새로운 장은 새로운 페이지에서 시작하지 않는다. \mainmatter 명령은 장절 번호붙임을 시작하고 페이지 번호를 아라비아 숫자로 붙인다. \backmatter 명령은 단지 장절 번호붙임만을 끈다. \tableofcontents 및 유사한 명령들, 그리고 \newlistof 명령에 의하여 만들어진 목차 명령은 언제나[1] \thispagestyle{chapter} 을 부른다. 만약 article 문서 옵션을 사용하고 있다면 *chapter* 페이지 스타일이 문서 전체의 일반적 모양과 동일하게 유지되게 하고 싶을 것이다.

표준 클래스에서와는 달리 ⟨*title*⟩은 raggedright로 조판된다. 만약 ⟨*title*⟩ 안에서 줄바꿈이 필요하다면 \\가 아니라 \newline을 써야 한다. 다음을 보라.

```
\section{A broken\newline title}
```

표준 클래스에서는 \section 들이 페이지 바닥에 너무 가까이 붙으면 다음 페이지의 첫 줄로 옮겨준다. 이런 일이 벌어지면 \flushbottom 명령이 효력을 발휘하게 되어 앞 페이지의 내용 마지막 줄이 편집 영역의 바닥까지 늘어나도록 간격이 설정된다.

```
\raggedbottomsectiontrue
\raggedbottomsectionfalse
\bottomsectionskip
```

\raggedbottomsectiontrue 선언은 장절 표제를 다음 페이지로 옮김으로써 발생하는 페이지 모자람을 \raggedbottom으로 식자하도록 설정한다. 그 외의 경우에는 영향을 미치지 않는다. \bottomsectionskip 길이는 페이지 모자람이 발생했을 때 허용되는 스트레치의 길이를 제어한다. 이 값을 0으로 하면 마지막 줄은 편집 영역의 바닥까지 늘려질 것이다. 기본값은 10mm이다.

\raggedbottomsectionfalse 선언은 이전의 \raggedbottomsectiontrue 선언을 취소한다. 이것이 기본값이다.

[1] 내부 매크로 호출 시점에 따른 결과이다.

\plainbreak{⟨*num*⟩} \plainbreak*{⟨*num*⟩}
\fancybreak{⟨*text*⟩} \fancybreak*{⟨*text*⟩}

\plainbreak는 장절 타이틀 없는 문서 구획이다. ⟨*num*⟩ 만큼의 빈 줄을 단락 사이에 넣고 그 다음 단락의 첫 줄을 들여밀기하지 않는다. 또하나의 무기명 문서구획으로 \fancybreak 명령이 있다. 이 명령은 ⟨*text*⟩를 단락 사이에 넣고 그 다음 단락의 첫 줄을 들여밀기하지 않는다. 예를 들면 다음과 같다.

```
\fancybreak{{*}\\{* * *}\\{*}}
```

위의 코드는 별표(asterisk)로 만든 다이아몬드를 문단 사이에 넣는다.

별표붙은 명령은 이어지는 단락의 첫 줄을 들여밀기하도록 만든다.

\plainfancybreak{⟨*space*⟩}{⟨*num*⟩}{⟨*text*⟩}
\plainfancybreak*{⟨*space*⟩}{⟨*num*⟩}{⟨*text*⟩}

plain break가 페이지의 처음이나 끝에 온다면 독자가 그것이 구획인지 여부를 알아보기가 매우 어렵다. \plainfancybreak는 페이지에 단락구획을 두고 그 뒤에 텍스트가 올 만한 충분한 공간이 있으면 ⟨*num*⟩ 행의 \plainbreak를 사용한다. 그렇지 못하고 단락 구획이 페이지의 첫머리나 끝에 오게되는 경우에는 ⟨*text*⟩의 \fancybreak를 사용한다. ⟨*space*⟩ 인자는 plain break 뒤에 ⟨*num*⟩ 개의 빈 줄과 몇 줄의 텍스트가 오기 위해 필요한 공간의 길이를 지정한다. 별표붙은 명령은 \plainbreak와 \fancybreak 명령의 별표붙은 버전을 사용하게 한다.

불행하게도 사용자가 요구한 plain, fancy break 공간 사이에는 상호간섭이 있다. P가 plain break를 위해서 요청된 공간(행)이고 F가 fancy break를 위해서 요구되는 공간(행)이라고 하자. S는 ⟨*space*⟩ 공간(행)이다. 몇 번의 실험을 통해, plain break 페이지의 위쪽과 아래쪽을 피할 조건은 $S-P>1$이라는 사실을 알게 되었다. 또한, fancy break가 페이지의 중간에 놓이지 않을 조건(즉, 상단과 하단이 아닌 위치에 올 조건)은 $S-F<3$이다. 예를 들어서, plain break와 fancy break가 동일한 수직 공간을 취하는 경우가 $S=P+2$이 만족되면 일어난다. 일반적으로 $F=P+n$이면 그 조건은 $1<S-P<3+n$이고, 이럴 경우에 \plainfancybreak 명령은 fancy break가 항상 plain break에 필요한 공간 만큼을 취하게 된다.

* * *

\plainfancybreak 매크로는 페이지의 중간에 plain break를 넣고 구획이 페이지의 처음이나 끝에 오면 fancy break를 넣는다.

\pfbreak \pfbreak*
\pfbreakskip
\pfbreakdisplay{⟨*text*⟩}

\pfbreak 매크로는 \plainfancybreak 대신 사용할 수 있는 좀더 편리한 매크로이다. plain break를 위한 공백은 \pfbreakskip 길이에 의하여 주어지는데, 이 값은 2행으로 초기화되어 있다. fancy break도 동일한 수직 공간을 취하며, \pfbreakdisplay의 ⟨*text*⟩ 인자로 주어져 있는 텍스트를 사용한다. 기본값 정의는 바로 위에서 보인 세 개의 별표를 찍는 것이다.

♣ ♢ ♣

\pfbreakdisplay 정의를 재설정함으로써 다른 모양을 사용할 수 있다. 여기 보인 fancy break와 같이 하려면 다음과 같이 정의한다.

```
\renewcommand{\pfbreakdisplay}{%
  $\clubsuit$\quad\$\diamondsuit$\quad$\clubsuit$}
\fancybreak{\pfbreakdisplay}
```

여기서는 \fancybreak를 사용하였다. 구획이 페이지 나눔이 될지 어떨지 알 수 없기 때문이다. \pfbreak라고만 하면 fancy display 대신 두 개 정도의 빈 줄을 넣어준다.

\pfbreak 명령 뒤에 이어지는 문단은 들여밀기되지 않는다. 들여밀기 하려면 \pfbreak* 명령을 쓴다.

```
\tableofcontents \tableofcontents*
\listoffigures \listoffigures*
\listoftables \listoftables*
```

표준 클래스에서 \tableofcontents 명령은 목차(ToC)를 문서의 주어진 위치에 식자한다. 반면, 이 클래스에서는 목차 자체도 ToC에 추가하도록 하고 있다. 별표붙은 명령인 \tableofcontents*는 목차 그 자체를 목차에 추가하지 않도록 한다. 그러므로 이 클래스의 \tableofcontents*가 표준 클래스의 \tableofcontents에 해당한다. 이 클래스는 또한 \listoffigures와 \listoftables의 경우에도 마찬가지로 '그림 목차'나 '표 목차' 자체를 ToC에 추가하는데, 별표붙은 명령을 쓰면 목차 제목을 ToC 엔트리에 넣지 않는다.

memoir 클래스 문서에서는 \tableofcontents나 \listoffigure 등을 한 번 이상 사용할 수 있다.

```
\appendix
\appendixname
```

표준 클래스에서 \appendix 명령은 chapter의 번호붙이기를 알파벳 형식으로 바꾸고 chaptername을 \appendixname (기본값은 Appendix)로 바꾼다. 따라서 \appendix 명령이 주어진 이후의 첫번째 \chapter 이후는 'Appendix A ...', 'Appendix B ...' 하는 식으로 나타나게 된다. 이 클래스는 이 이외에도 부록을 다루는 다른 방법을 더 제공한다.

【memhangul】 한글 문서에서 \appendixname은 '부록'이다. 따라서 '부록 A', '부록 B' 형식으로 장 제목 번호가 짜여진다. 그러나 일반적인 장은 '제 1 장', '제 2 장' 형식이므로 \chaptername과 \appendixname은 장 카운터에 대하여 놓이는 위치가 다르다. 이를 올바르게 구현하기 위해서는 장 제목 설정을 상당히 주의해서 제어해야 한다. memhangul-ucs는 이러한 일이 별다른 조치없이 구현될 수 있도록 하고 있지만, 사용자가 이 설정을 바꿀 때는 몇 가지 주의가 필요하다. ■

\appendixpage
\appendixpage*
\appendixpagename

\appendixpage 명령은 part와 같은 형식(그러나 편명 Part나 번호는 붙지 않는다)의 페이지를 만들어서 \appendixpagename (기본값은 Appendices) 으로 지정된 타이틀을 식자한다. 또 \addappheadtotoc 명령을 이용하여 ToC에 차례 항목을 추가한다. 별표붙은 명령은 appendix page를 만들지만 차례 항목을 추가하지 않는다.

\addappheadtotoc
\appendixtocname

\addappheadtotoc 명령은 ToC에 항목을 추가한다. \appendixtocname (기본값은 'Appendices')로 주어지는 타이틀을 사용한다.

\begin{appendices} text \end{appendices}

appendices 환경은 \appendix 명령과 비슷하게 동작한다. 즉 번호붙임과 장의 제목을 바꾼다. 그러나 이 환경이 종료되면 chapter는 원래 값으로 복귀하고 장의 번호붙임은 appendices 환경과 관계없이 이전에서부터 이어진다.

\begin{subappendices} text \end{subappendices}
\namesubappendixtrue \namesubappendixfalse

subappendices 환경은 장의 끝에 붙이는 부록이다. 이 환경 안에서는 \section이 새로운 sub-appendix로 시작한다. \addappheadtotoc를 환경이 시작하는 곳에 두면 ToC에 목차 항목으로 들어가게 된다. \namesubappendixtrue 선언을 subappendices 환경 앞에 두면 문서 본문의 sub-appendix 번호가 \appendixname 값 앞에 놓인다. \namesubappendixfalse 선언은 이 동작을 끈다.

표 8.1: 문서의 장절구분 레벨

Division	Level
\part	-1
\chapter	0
\section	1
\subsection	2
\subsubsection	3
\paragraph	4
\subparagraph	5

8.4 번호붙이기

각 장절명령은 표 [8.1]에 보인 바와 같이 *level* 값이 부여되어 있다. secnumdepth 카운터 값이 이 레벨값과 같거나 크다면 그 장절명령에는 번호가 붙는다. 예를 들면,

```
\setcounter{secnumdepth}{2}
```

위와 같이 하면 subsection에서 part까지가 번호붙는 장절표제가 된다.

\setsecnumdepth{⟨*secname*⟩}
\maxsecnumdepth{⟨*secname*⟩}

이 level 숫자를 기억하는 것은 귀찮은 일이므로, \setsecnumdepth 명령을 이용하여 장절 명령의 이름으로 편리하게 지정할 수 있다. 이 명령은 secnumdepth을 ⟨*secname*⟩을 지정함으로써 설정해준다. ⟨*secname*⟩은 백슬래시를 붙이지 않은 장절명령 이름이다.예를 들어 subsection까지 숫자를 붙이고 싶다면 다음과 같이 한다.

```
\setsecnumdepth{subsection}
```

\maxsecnumdepth 명령은 문서에서 사용할 secnumdepth의 최대값을 지정한다. 문서의 어떤 곳에서든 (임시로) (임시로) 번호붙임 수준을 바꾸기 위해서 \setsecnumdepth 명령을 사용할 수 있다.

이 클래스의 기본값은 다음과 같다.

```
\maxsecnumdepth{section}
\setsecnumdepth{section}
```

\mainmatter와 \mainmatter* 명령이 불릴 때 secnumdepth를 \maxsecnumdepth 값으로 설정한다.

번호붙임 설정 명령들은 tocvsec2 패키지 [Wil99b] 에서 가져왔다.

8.5 Part 헤딩

Part 타이틀은 언제나 새로운 페이지를 시작하고 *part* 페이지스타일을 적용한다.

\part 타이틀의 조판 방식은 설정가능하다.

\beforepartskip \afterpartskip

위의 두 개의 명령은 part title의 전후에 올 공백을 효과적으로 제어한다. 기본 정의는 다음과 같다.

```
\newcommand{\beforepartskip}{\null\vfil}
\newcommand{\afterpartskip}{\vfil\newpage}
```

이 정의에 의해서 part 타이틀은 페이지의 수직 중앙정렬되고 새로운 페이지가 시작된다. 타이틀을 페이지의 위쪽으로 이동시키려면 예컨대 다음과 같이 한다.

```
\renewcommand{\beforepartskip}{\null\vskip 0pt plus 0.3fil}
\renewcommand{\afterpartskip}{\vskip 0pt plus 0.7fil \newpage}
```

\midpartskip

\midpartskip 길이는 part 번호행과 part 타이틀 행 사이의 수직 간격을 나타낸다. 기본값은 20pt인데 \setlength나 \addtolength 명령으로 바꿀 수 있다. \baselineskip에 대한 상대값을 설정하면 폰트 크기에 따라 이 값이 조절될 것이기 때문에 고정값을 이용하는 것보다 낫다.

\printpartname \partnamefont \partnamenum \printpartnum \partnumfont

\printpartname 매크로는 \partnamefont에 의해 설정된 폰트로 part 이름(\partname 매크로로 지정된 문자열)을 식자한다. 기본값은 \huge 사이즈의 \bfseries 폰트를 이용하는 것이다. 마찬가지로 part 번호는 \printpartnum으로 식자되며, 이 때 \partnumfont로 지정된 폰트를 이용한다. \partnamenum 매크로는 part 이름과 part 번호 사이에 오게 되는 것으로 기본값은 space로 정의되어 있다. 이 설정들을 바꾸면 효과가 나타난다.

```
\renewcommand{\partnamefont}{\normalfont\huge\sffamily\raggedright}
\renewcommand{\partnumfont}{\normalfont\huge\sffamily}
```

part 번호를 기본 폰트로만 나타내고 싶다면 다음과 같이 한다.

```
\renewcommand{\printpartname}{}
\renewcommand{\partnamenum}{}
```

【memhangul】 memhangul-ucs 패키지는 한글 문서의 편번호제목을 제대로 식자하기 위해서 다음과 같은 매크로를 추가하였다. \printpartname 매크로는 사용하지 않는다.

\prepartnum \postpartnum \hparttitlehead

\prepartnum은 "제", \postpartnum은 \partname, \hparttitlehead는 "제 \thepart 편"으로 정의되어 있다. \hparttitlehead는 ToC와 난외표제에 써야할 경우에 사용한다. 만약 "제"와 "편" 없이 그냥 I.로만 나타내고 싶으면,

```
\renewcommand{\prepartnum}{}
\renewcommand{\postpartnum}{.}
\renewcommand{\partnamenum}{}
```

위와 같이 정의한다. ■

\printparttitle{⟨*title*⟩} \parttitlefont

타이틀은 \printparttitle로 식자되며, 이 때 글꼴은 \parttitlefont로 지정된다. 기본값은 \Huge 크기의 \bfseries 폰트이다. 이것을 예컨대 small caps 폰트로 raggedleft 하려면 다음과 같이 지정한다.

```
\renewcommmand{\parttitlefont}{\normalfont\Huge\scshape\raggedleft}
```

\parttitlefont 글꼴은 부록을 조판할 때 \appendixpage와 그 별표붙은 명령에도 사용된다.

8.6 chapter 헤딩

chapter 헤딩은 part 헤딩과 거의 유사하게 설정할 수 있다. 그러나 이에 더하여 내장 chapter 스타일을 사용할 수도 있고 자신만의 스타일을 정의할 수도 있다.

chapter는 항상 새로운 페이지를 시작하고 *chapter* 페이지스타일을 적용한다. 새로 시작하는 페이지가 홀수쪽이 되는가 짝수쪽이 되는가는 클래스 옵션으로 지정한다. oneside 옵션을 썼다면 바로 다음 페이지에서 시작하지만 twoside 옵션의 경우에는 시작하는 위치가 다음과 같이 정해져 있다.

openright chapter 헤딩은 다음번 홀수쪽에 식자되고 그 이전 짝수쪽의 남은 공간은 비운다.

openleft chapter 헤딩은 다음번 짝수쪽에 식자되고 그 이전 홀수쪽의 남은 공간은 비운다.

openany chapter 헤딩은 다음 페이지에 식자되고 페이지를 비워넘기지 않는다.

```
\openright \openleft \openany
```

이 세 개의 선언은 같은 이름을 가진 문서 옵션과 동일한 효과를 낸다. 이 선언을 둠으로써 장의 시작 페이지를 문서 어디에서든지 바꿀 수 있다.

```
\clearforchapter
```

장의 시작 페이지를 실제 만드는 매크로는 \clearforchapter이다. \openright, \openleft, \openany 명령은 \clearforchapter를 각각 \cleartorecto, \cleartoverso, \clearpage로 정의한다. \clearforchapter를 새로운 페이지를 시작하는 것 이상의 다른 기능을 갖도록 정의하는 것도 가능하다.

어떤 책을 보면 왼쪽 페이지에 삽화나 격언 등을 넣은 다음 오른쪽 페이지부터 텍스트를 시작하는 경우도 있다. 이런 효과를 얻으려면 다음과 같이 한다.

```
\openleft               % chapter title on verso page
\chapter{The title}     % chapter title
\begin{centering}       % include a centered illustration
\includegraphics{...}
\end{centering}
\clearpage              % go to recto page
Start of the text       % chapter body
```

```
\chapterstyle{⟨style⟩}
```

\chapterstyle 매크로는 \pagestyle 명령과 유사한 것이지만 이후의 chapter 헤딩을 ⟨*style*⟩ 형태로 설정하는 데 쓰인다.

이 클래스는 미리 정의된 chapter 스타일을 몇 가지 제공한다. *default*는 LaTeX의 book 클래스에서 chapter 헤딩 스타일로 제공하는 익숙한 것이다. 예컨대 *fred* chapter 스타일을 사용하려면 \chapterstyle{fred}라고 지시한다. 하나의 문서에 서로 다른 chapter 스타일을 사용할 수 있다. 기정의 스타일은 다음과 같다.

default LaTeX의 기본 스타일.

section 헤딩은 section처럼 식자된다. 즉, 번호만이 표시되고 타이틀과 번호표제가 같은 줄에 놓인다. 이 스타일의 예제는 129 쪽의 제 10 장에서 볼 수 있다.

hangnum *section* 스타일과 비슷하지만 번호표제가 마진 영역에 온다. 이 스타일의 예제는 159 쪽의 제 11 장에서 볼 수 있다.

companion chapter 헤딩은 *LaTeXCompanion* 시리즈의 것과 비슷하다. 보기는 227 쪽의 제 14 장를 보라.

article 헤딩은 article 클래스의 `section` 헤딩과 같다. 위의 *section* 스타일과 비슷하지만 폰트와 간격주기가 다르다.

demo 이 스타일은 시험용으로 작성된 것이다. 243 쪽의 제 15 장에서 어떤 모양인지 확인할 수 있다.

```
\beforechapskip \afterchapskip
```

이 두 개의 길이는 chapter 헤딩의 전후 간격을 설정하는 것이다. 기본값은 각각 50pt와 40pt이다.

```
\midchapskip
```

길이 `\midchapskip`은 장 번호행과 장 타이틀 사이의 수직 간격을 나타낸다. 기본값은 20pt인데 `\setlength`나 `\addtolength`로 바꿀 수 있다. 간격을 `\baselineskip`을 기준으로 설정하면 선택된 폰트 크기에 대하여 상대적으로 변하기 때문에 고정 값을 쓰는 것보다 나을 수 있다.

```
\printchaptername \chapnamefont
\chapternamenum
\printchapternum \chapnumfont
```

`\printchapername` 매크로는 장 이름 (`\chaptername`에 의해 주어지는 것으로 기본값은 `Chapter`이다) 을 `\chapnamefont`로 설정된 폰트를 사용하여 식자한다. 기본값은 `\huge` 사이즈의 `\bfseries` 폰트이다. 장 번호는 `\chapnumfont`로 설정된 폰트를 이용하여 `\printchapternum`으로 식자한다. 기본값은 `\chapnamefont`와 동일하다. `\chapternamenum` 매크로는 장 이름과 번호 사이의 공백을 설정하는 것인데 처음에는 space 하나로 정의되어 있다.

【memhangul】 memhangul-ucs에서는 part의 경우와 같이 장번호 제목을 제대로 식자하기 위해 매크로를 추가하였다. `\printchaptername` 매크로는 사용하지 않는다.

```
\prechapternum \postchapternum
\hchaptertitlehead
```

기본적으로 `\prechapternum`은 '제', `\postchapternum`은 `\chaptername`, `\hchaptertitlehead`는 '제 `\thechapter` 장'의 형식으로 설정되어 있다. `\hchaptertitlehead`는 ToC와 난외표제에 사용된다. 사용자가 새로운 chapter 스타일을 정의할 때는 적어도 `\prechapternum`과 `\postchapternum`을 재설정하여야 하는 경우가 많음에 유의하라. 예컨대, 이 장의 경우는 원래의 영문판 설정에,

```
\renewcommand{\prechapternum}{}% <= 이 명령을 정의. 여기서는 비움.
\renewcommand{\postchapternum}{}% <= 이 명령을 정의. 여기서는 비움.
```

이 두 행을 추가하였다. ■

\printchaptertitle{⟨*title*⟩} \chaptitlefont

\printchaptertitle은 \chaptitlefont에 정의된 폰트로 타이틀을 식자한다. 기본값은 \Huge 사이즈의 \bfseries 폰트로 설정되어 있다.

\insertchapterspace

\chapter 명령은 그림 목차나 표 목차에서 약간의 수직 간격을 삽입한다. 이 때 불리는 매크로는 \insertchapterspace인데 기본 정의는 다음과 같다.

```
\newcommand{\insertchapterspace}{%
  \addtocontents{lof}{\protect\addvspace{10pt}}%
  \addtocontents{lot}{\protect\addvspace{10pt}}%
}
```

만약 장이 바뀔 때의 공간을 넣고 싶지 않다면 \renewcommand{\insertchapterspace}{} 하면 된다. 이 공백은 \addvspace에 길이값 인자를 줌으로써 바꿀 수 있다.

chapter 스타일 정의하기

다음 코드는 번호붙는 장 헤딩을 조판하기 위해 필요한 것들을 보여준다.

```
\chapterheadstart
\printchaptername \chapternamenum \printchapternum
\afterchapternum
\printchaptertitle{The chapter title}
\afterchaptertitle
```

\printchapternonum

번호붙지 않는 장제목에 필요한 것은 좀더 단순하다.

```
\chapterheadstart
\pintchapternonum
\printchaptertitle{The chapter title}
\afterchaptertitle
```

이 각각의 요소들은 처음에 다음과 같이 정의되어 있다.

```
\newcommmand{\chapterheadstart}{\vspace*{\beforechapskip}}
\newcommand{\printchaptername}{\chapnamefont \@chapapp}
\newcommand{\chapternamenum}{\space}
\newcommand{\printchapternum}{\chapnumfont \thechapter}
\newcommand{\afterchapternum}{\par\nobreak\vskip \midchapskip}
\newcommand{\printchapternonum}{}
\newcommand{\printchaptertitle}[1]{\chaptitlefont #1}
\newcommand{\afterchaptertitle}{\par\nobreak\vskip \afterchapskip}
```

【memhangul】 한글 문서에서는 여기서 \printchaptername 매크로가 무의미하고 \prechapternum과 \postchapternum 매크로 정의를 추가하여야 한다. 기본값은 다음과 같이 되어 있다.

```
\newcommand{\prechapternum}{\chapnamefont \pre@chapter}
\newcommand{\postchapternum}{\chapnamefont \post@chapter}
```

\pre@chapter와 \post@chapter는 각각 '제'와 '장'을 나타내는 내부명령이다. ■

새로운 스타일을 만들려면 위의 매크로 정의들과 폰트 및 간격을 재설정하면 된다.

\makechapterstyle{⟨*style*⟩}{⟨*text*⟩}

\makechapterstyle 명령은 새로운 장 스타일을 만드는 데 쓰인다. ⟨*style*⟩은 정의하고자 하는 스타일의 이름이고 ⟨*text*⟩는 스타일을 설정하는 LaTeX 코드들이다.

예로서, *section* 장 스타일 정의 코드를 살펴보자.

```
\makechapterstyle{section}{%
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\chapnumfont}{\chaptitlefont}
  \renewcommand{\printchapternum}{\chapnumfont \thechapter\space}
  \renewcommand{\afterchapternum}{}
}
```

여기서 \printchaptername 비어 있다. 그러므로 'Chapter'라는 문자열은 식자되지 않는다. 번호행과 타이틀에 동일한 폰트가 사용되었다. 그리고 번호 뒤에 space 하나가 따라 붙는다. \afterchapternum 매크로는 비어 있다. 그러므로 chapter title은 숫자 바로 뒤에 위치한다.

표준 클래스에서 번호붙지 않는 chapter는 번호 붙는 chapter의 'Chapter'라는 단어가 식자되는 곳과 같은 위치에 온다. \printchapternonum 매크로는 번호붙지 않은 chapter 타이틀 텍스트가 식자되기 직전에 호출된다. 기본값은 아무것도 하지 않는 것이지만 원한다면 \renewcommand로 이것을 수정할 수 있다. 예를 들면, 번호붙은 장과 번호

붙지 않은 장의 타이틀이 둘다 주어진 페이지의 동일한 수직 위치에 오도록 하고 싶다면, `\printchapternonum`에 'Chapter N'에 해당하는 수직 공간을 넣어주도록 재정의할 수 있다.

Bastiaan Veelo는 2003년 7월 22일에 *[memoir] [contrib]* 새 장 스타일이라는 제목으로 CTT에 새로운 chapter 스타일 코드를 기고하였다. 그의 코드를 조금 바꾸어서 이름을 *veelo*라고 붙였다. 아래 그 코드가 있다. 내가 약간의 편집적 수정을 가했다.

> 새로운 장 스타일을 memoir 클래스와 공유하고자 합니다. 이 스타일은 폭이 조금 좁은 문서에 적합하도록 짜여졌습니다. 문서를 접어서 재단하면 검은색 탭이 새 장이 시작하는 바깥쪽에 표시되기 때문에 장의 위치를 쉽게 찾을 수 있게 합니다. 장 번호를 확대하였으므로 대부분의 DVI viewer 프로그램에서는 제대로 보이지 않습니다.
>
> Bastiaan

```
\makeatletter
\newlength{\numberheight}
\newlength{\barlength}
\makechapterstyle{veelo}{%
   \setlength{\beforechapskip}{40pt}
   \setlength{\midchapskip}{25pt}
   \setlength{\afterchapskip}{40pt}
   \renewcommand{\chapnamefont}{\normalfont\LARGE\flushright}
   \renewcommand{\chapnumfont}{\normalfont\HUGE}
   \renewcommand{\chaptitlefont}{\normalfont\HUGE\bfseries\flushright}
   \renewcommand{\printchaptername}{%
     \chapnamefont\MakeUppercase{\@chapapp}}
   \renewcommand{\chapternamenum}{}
   \setlength{\numberheight}{18mm}
   \setlength{\barlength}{\paperwidth}
   \addtolength{\barlength}{-\textwidth}
   \addtolength{\barlength}{-\spinemargin}
   \renewcommand{\printchapternum}{%
     \makebox[0pt][l]{%
       \hspace{.8em}%
       \resizebox{!}{\numberheight}{\chapnumfont \thechapter}%
       \hspace{.8em}%
       \rule{\barlength}{\numberheight}
     }
   }
   \makeoddfoot{plain}{}{}{\thepage}
}
\makeatother
```

【memhangul】 memhangul-ucs0에서 *veelo*를 사용할 때, 위와 다른 부분만 제시하면 다음과 같다.

```
%\renewcommand{\printchaptername}{} % 한글버전에서는 disable.
\renewcommand{\prechapternum}{%
 \chapnamefont\MakeUppercase{chapter}}% <= 이 명령을 정의.
\renewcommand{\postchapternum}{}% <= 이 명령을 정의. 여기서는 비움.
```

\printchaptername 명령 대신 \prechapternum과 \postchapternum을 정의하고 있음에 주의하라. ■

위의 설정이 효력을 가지려면 graphicx 패키지 [CR99]가 필요하다. \resizebox 명령이 사용되고 있기 때문이다. *veelo* 스타일은 홀수쪽으로 장이 시작할 때 가장 잘 작동한다.

그밖의 장 스타일의 정의에 대해서는 클래스 코드 문서를 살펴보라. 새로운 자신의 스타일을 작성하려 할 때 도움이 될 것이다.

장 헤딩을 바꾸려면 굳이 새로운 장 스타일을 정의할 필요는 없다는 데 주의하라. 각각의 매크로를 개별적으로 수정하면 되고 굳이 새로운 스타일에 넣지 않아도 된다.

8.7 하위 수준 장절명령

하위 수준의 장절 헤딩—section에서 subparagraph까지—들도 설정가능하다. 그러나 chapter 스타일과는 방식이 다르다.

고려해야 할 점이 세 가지 있다. (a) 장절표제행 이전 텍스트의 베이스라인과 장절표제행 베이스라인 사이의 수직 거리, (b) 표제행의 왼쪽 마진에서부터의 들여밀기 값, (c) 장절 제목 표제에 적용될 스타일과 폰트. 그리고, 장절 제목이 문단 첫머리(run-in)에 오도록 할 것인지 별행으로 할 것인지를 고려해야 한다. 별행으로 하는 경우에는 제목과 본문 사이의 수직 간격도 적절하게 조절되어야 한다. 그림 (8.1)은 별행 절 표제행을 제어하는 파라미터들을 보여주고, 그림 (8.2)는 문단 첫머리 절 표제에 해당하는 파라미터들을 보여준다.

이제부터 S라는 부호로써 sec, subsec, subsubsec, para, subpara를 나타내도록 하겠다. 각각 section에서부터 subparagraph까지 명칭을 줄여서 나타낸 것이다.

\setbeforeSskip{⟨*skip*⟩}

⟨*skip*⟩ 길이 인자의 절대값은 헤딩의 윗쪽 공백 크기를 나타낸다. 이 값이 음수이면 헤딩 이후의 단락의 첫번째 줄이 들여밀기 되지 않는다. ⟨*skip*⟩의 기본값은 장절명령의 레벨에 따라 달라지는데, \section의 경우, 즉 S = sec인 경우 다음과 같다.

```
-3.5ex plus -1ex minus -.2ex
```

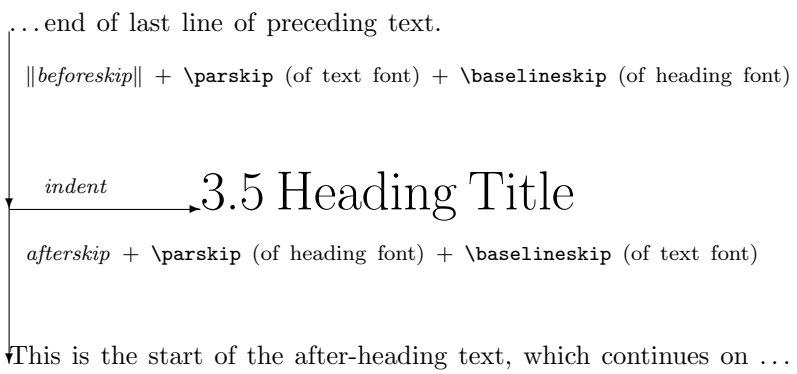


그림 8.1: 별행 절 헤딩

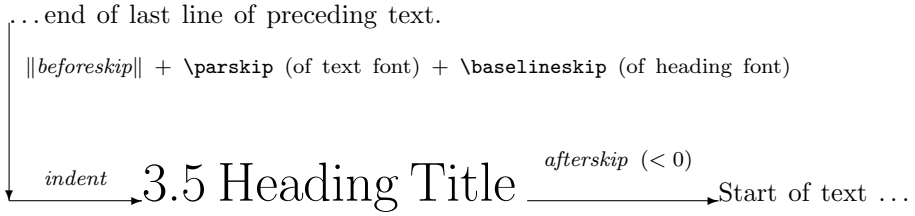


그림 8.2: 문단첫머리 절 헤딩

plus와 minus 값은 간격의 늘림·줄임 허용치를 가리킨다. 모든 값이 음수이므로 이어지는 단락은 들여밀기가 되지 않을 것이다. 들여밀기를 허용하고 싶다면,

```
\setbeforesecskip{3.5ex plus 1ex minus .2ex}
```

위와 같이 설정한다.

`\setSindent{`⟨*length*⟩`}`

⟨*length*⟩ 길이 인자의 값으로 표제(절번호와 제목) 들여밀기를 지정한다. 들여밀기의 기준점은 왼쪽 마진의 끝이다. 보통 0pt이다.

\setSheadstyle{⟨*text*⟩}

이 매크로는 절 번호와 제목에 해당하는 스타일과 폰트를 지정한다. 앞서와 마찬가지로 ⟨*text*⟩ 인자의 기본값은 장절표제의 레벨에 따라 달라진다. \subsection, 즉 S = subsec의 경우 이것은 \large\bfseries\raggedright로서, \large 크기의 \bfseries 폰트를 사용한다. 그리고 타이틀은 ragged right로 정렬되므로 여러 줄 제목의 경우 하이픈 적용 없이 단어단위로 행을 끊는다.

⟨*text*⟩ 인자의 제일 마지막 요소는 한 개의 인자를 취하는 매크로를 쓸 수 있다. 이 때 한 개의 인자 자리에는 절 번호와 절 제목이 들어간다. 그러므로 \subsubsection 타이틀을 모두 대문자로 하고 중앙정렬, normal font로 식자하고 싶다면, 다음과 같이 한다.

```
\setsubsubsecheadtstyle{\normalfont\centering\MakeUppercase}
```

또다른 보기로서, 별로 권장할 만한 것은 아니지만 절 제목 밑에 선을 그을 수도 있다.

```
\newcommand{\ruledsec}[1]{%
  \Large\bfseries\raggedright #1 \rule{\textwidth}{0.4pt}}
\setsecheadstyle{\ruledsec}
```

\setafterSskip{⟨*skip*⟩}

⟨*skip*⟩ 길이 인자 값이 양수라면 별행 표제와 이어지는 텍스트 사이에 공백이 벌어진다. 이 값이 음수라면 표제는 문단첫머리에 놓이고 제목과 이어지는 텍스트 사이가 주어진 값만큼 수평 간격을 두게 된다. ⟨*skip*⟩의 기본값은 장절명령의 레벨에 따라 달라지는데, \section의 경우, 즉 S = sec일 때 이 값은 2.3ex plus .2ex이다. \subparagraph, 즉 S = subpara일 때는 문단첫머리 표제로 식자되는데, 기본값은 -1em이다.

\@hangfrom{⟨*stuff*⟩}
\sethangfrom{⟨*code*⟩}

내부적으로 모든 타이틀만들기 매크로는 \@hangfrom 매크로를 호출한다. 이 매크로는 여러 줄 타이틀을 내어밀기 단락처럼 보이도록 만든다. \@hangfrom의 기본 정의는 ltsect.dtx 파일에 나와있는데, 다음과 같이 정의되어 있다.

```
\newcommand{\@hangfrom}[1]{\setbox\@tempboxa\hbox{{#1}}%
  \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

하나의 박스 안에 인자를 넣고 그 폭을 측정한 다음, 들여밀기 문단을 만든다. 첫번째 줄은 두번째 이후의 줄에 비해서 인자의 폭(width) 만큼 내어밀기된다.

\sethangfrom 매크로는 \@hangfrom 매크로를 ⟨*code*⟩로 재설정한다. 예를 들어서 타이틀을 내어밀기 문단이 아닌 직사각형 문단으로 설정하려 할 때는 간단히 다음과 같이 한다.

```
\sethangfrom{\noindent #1}
```

이 명령을 쓸 때 주의할 점은 \@hangfrom에 넘길 인자가 놓일 위치를 표시하기 위해서 대체할 코드 안에 #1를 써주어야 한다는 점이다.

\@seccntformat{⟨*stuff*⟩}
\setsecnumformat{⟨*code*⟩}

내부적으로 모든 타이틀만들기 매크로는 \@seccntformat이라는 매크로를 호출한다. 이 매크로는 타이틀 내의 절번호 형식을 지정하는 데 사용된다. 기본 정의는 ltsect.dtx 파일에 들어 있는데, 다음과 같다.

```
\newcommand{\@seccntformat}[1]{\csname the#1\endcsname\quad}
```

위의 정의에 따르면 \thesec...으로 시작하는 절번호 형식은 뒤에 스페이스를 두고 있다. \setsecnumformat 명령은 \@seccntformat을 ⟨*code*⟩로 재정의하게 하는 역할을 한다. 예를 들면 절번호 뒤에 콜론과 스페이스를 두게 하려는 경우를 보자.

```
\setsecnumformat{\csname the#1\endcsname:\quad}
```

\@seccntformat에 넘겨줄 인자(절 번호)를 표시하기 위해 #1를 사용해야 한다는 점에 주의하라.

\hangsecnum
\defaultsecnum

\hangsecnum 매크로는 절번호가 마진쪽으로 내어밀기 되도록 선언한다. \defaultsecnum 매크로는 \hangsecnum 선언에 의해 만들어진 내어밀기 설정을 되돌리는 선언이다. 즉, 절번호가 익숙한 위치에 놓이도록 한다.

\Shook
\setShook{⟨*text*⟩}

Shook 매크로는 타이틀을 조판하기 직전에 호출되는데, 기본값은 아무것도 하지 않는 것이다. \setShook 매크로를 이용하여 \Shook을 ⟨*text*⟩ 코드로 재정의할 수 있다. 예를 들자면, \sethangfrom이나 \setsecnumformat 명령을 특정 절구획 명령 정의에 포함시키고자 할 때, 이 매크로를 사용할 수 있다.

【memhangul】 memhangul-ucs 패키지는 \section 하위 명령에 대해서는 별도의 조치를 취하고 있지 않다. 그러므로 아래 memoir 클래스의 section 하위 명령 설정을 원하는 대로 적용하면 된다. ■

표 8.2: 장절명령 헤딩의 기본 폰트

\partnamefont	\huge\bfseries
\partnumfont	\huge\bfseries
\parttitlefont	\Huge\bfseries
\chapnamefont	\normalfont\huge\bfseries
\chapnumfont	\normalfont\huge\bfseries
\chaptitlefont	\normalfont\Huge\bfseries
\secheadstyle	\Large\bfseries\raggedright
\subsecheadstyle	\large\bfseries\raggedright
\subsubsecheadstyle	\normalsize\bfseries\raggedright
\paraheadstyle	\normalsize\bfseries
\subparaheadstyle	\normalsize\bfseries

8.8 이 장의 헤딩 스타일

표 [8.2]에는 각 장절명령 표제의 기본 폰트가 나열되어 있다. 이 폰트는 모두 bold로, 장절명령의 레벨에 따라 다른 크기로 식자하도록 설정되어 있다.

아래 보인 것은 이 장이 시작하기 이전에 설정한 것이다.

이 장에서는 bold 글꼴 대신 sans 글꼴을 사용하였다. chapter에서 subsubsection까지 아래 명령 설정을 보였다.

```
\renewcommand{\chapnumfont}{\normalfont\huge\sffamily}
\renewcommand{\chaptitlefont}{\chapnumfont}
\setsecheadstyle{\Large\sffamily\raggedright}
\setsubsecheadstyle{\large\sffamily\raggedright}
\setsubsubsecheadstyle{\normalsize\sffamily\raggedright}
```

장의 표제는 section 표제처럼 식자된다. 이를 위해 \printchaptername과 \chapternamenum이 아무런 역할을 하지 않도록 재정의했다. 이렇게 하면 장 이름이 표시되지 않고 장 이름과 장 번호 사이에 아무것도 추가로 오지 않는다. \afterchapternum은 그저 \space만을 둠으로써 보통 장번호와 장제목 사이에 오는 수직 간격을 없앴다. 그 대신 장 번호와 장제목 사이에 단순한 space 하나가 오도록 한 것이다.

```
\renewcommand{\printchaptername}{}
\renewcommand{\chapternamenum}{}
\renewcommand{\afterchapternum}{\space}
```

【memhangul】 한글 번역본에서는 \printchaptername을 무시한다. 그리고,

```
\renewcommand{\prechapternum}{}% <= 이 명령을 정의. 여기서는 비움.
\renewcommand{\postchapternum}{}% <= 이 명령을 정의. 여기서는 비움.
```

위와 같은 정의를 더 두었다. ■

다음 장이 시작하기 전에 기본 장절명령 형식으로 되돌아가려 한다. 그래서, 다음 코드를 이 장의 마지막에 두었다.

```
\chapterstyle{default}
\setsecheadstyle{\Large\bfseries\raggedright}
\setsubsecheadstyle{\large\bfseries\raggedright}
\setsubsubsecheadstyle{\normalsize\bfseries\raggedright}
```

`\chapterstyle{default}`를 호출하면 장 표제에 영향을 미친다. 다른 세 개의 명령은 하위 장절명령 글꼴을 기본값으로 되돌리는 것이다.

8.9 각주와 면주

장절명령의 인자 ⟨*title*⟩은 문서 내에서 절표제로 사용된다.

그런데, 옵션 인자로 ⟨*toc-title*⟩을 줄 때는, 장절 명령의 인자 ⟨*title*⟩은 고정되어 있더라도, 풀리는 명령들을 `\protect`해주어야 한다. ⟨*toc-title*⟩은 또 두 가지 역할을 한다.

1. ToC에 들어갈 타이틀 텍스트로 쓰인다.
2. 면주에서 사용되는 텍스트로 쓰인다.

옵션 인자가 주어지지 않으면 ⟨*title*⟩이, 본문에 쓰일 장절표제, ToC에 들어가는 타이틀, 페이지 헤더에 쓰일 타이틀, 세 가지 역할을 수행하게 된다.

어떤 사람은 장절 타이틀에 각주를 꼭 붙이고 싶어한다. 되도록이면 이런 일은 하지 않는 것이 좋다. 그러나 아무래도 그렇게 하고 싶으면 반드시 옵션 인자를 사용하여 `\footnote` 명령을 본문의 장절표제에만 붙어야 할 것이다. 만약 옵션 인자를 사용하지 않게 되면 각주 표지와 각주 텍스트가 장절 표제 페이지와 ToC, 그리고 헤더에 포함되는 ⟨*title*⟩을 포함한 여러 페이지에 난잡하게 흩어져서 나타나게 될 것인데, 이것은 누구도 바라는 상황이 아니다. 그러므로, 장절 타이틀에 꼭 각주를 붙여야 한다면, 다음과 같이 하라.

```
\chapter[Title text]{Title text\footnote{Do you really have to do this?}}
```

8.10 페이지매김과 면번호

LaTeX 문서에서는, 페이지가 나누어질 때 페이지 번호매김이 이루어진다. 즉, 각각의 페이지에 번호가 할당되고 이 번호는 page 카운터의 값이다. 이 값은 `\setcounter`나 `\addtocounter` 명령으로 언제라도 바꿀 수 있다.

\pagenumbering{⟨*num-style*⟩} \pagenumbering*{⟨*num-style*⟩}

\pagenumbering과 \pagenumbering* 매크로는 면번호(folio) 위치에 ⟨*num-style*⟩을 이용하여 페이지 번호를 찍도록 한다. ⟨*num-style*⟩은 Alph, alph, arabic, Roman, roman 가운데 하나인데, 각각 알파벳 대문자, 소문자, 아라비아 숫자, 대문자 로마숫자, 소문자 로마숫자를 의미한다. 영문자는 26 글자밖에 없기 때문에 Alph나 alph는 몇 페이지 정도에만 쓸 수 있다. 당연히 이 매크로는 \thepage가 \num-style{page} 형식이 되도록 재정의한다. 또한, \pagenumbering 매크로는 page 카운터를 1로 재설정한다. 별표붙은 명령은 카운터 재설정을 하지 않는다.

페이지 번호붙이기 스타일이 바뀔 때는 페이지 번호를 되돌려서 재설정하는 것이 일반적이다. 그러나 페이지 번호 형식과는 상관없이 번호가 일련번호로 매겨지면 좋은 경우가 있기 때문에, 이 때는 \pagenumbering* 명령이 편리하다.

\savepagenumber \restorepagenumber

\savepagenumber 매크로는 현재의 페이지 번호를 저장한다. \restorepagenumber 매크로는 저장되었던 값으로 되돌려준다. 이 한 쌍의 명령은 페이지 번호매김을 잠시 중단하고 싶을 때 쓸 수 있다. 예를 들면, 몇 페이지에 걸쳐서 별지 삽화면을 삽입하려 하는 경우, 전자 문서에서는 페이지 번호매김이 불필요하다. 이럴 경우에는 다음과 같이 하면 된다.

```
\clearpage          % get onto next page
\savepagenumber     % save the page number
\pagestyle{empty}   % no headers or footers
%% insert the illustrations
\clearpage
\pagestyle{...}
\restorepagenumber
...
```

복구된 페이지 번호가 적절한지 살펴보고 필요하다면 1 정도를 증가시키거나 감소시켜야 할 때도 있다.

```
\restorepagenumber
% perhaps \addtocounter{page}{1} or \addtocounter{page}{-1}
```

이것은 \...pagenumber 명령이 실행된 위치에 좌우되는 것으로서 TEX이 페이지 나눔을 실행하기 전이냐 후이냐에 달려 있다.

제 9 장

문단과 리스트

이 장은 default 장절 표제 스타일로 조판되었다.

9.1 서론

장절 구분 범위 내의 텍스트는 문단으로 나누어진다. 때때로 일반적인 텍스트의 흐름으로부터 구분되는 인용문이나 항목 나열과 같은 텍스트가 있을 수 있다.

9.2 문단

표준 문단의 모양을 제어하는 두 개의 기본적인 매개변수가 있다.

```
\parindent \parskip
```

길이값 \parindent는 문단의 첫 줄의 들여밀기 값이다. 길이값 \parskip은 문단과 문단 사이의 수직 간격이다. 이들은 그림 (9.1)에 그림으로 나타내었다. \parskip 값은 일반적으로 0pt이며, \parindent는 대개 사용되고 있는 기본 폰트에 따라 실제 들여쓰기 값이 달라지도록 em단위로 정의되어 있다. 만약 \parindent가 음수로 지정되면 문단의 첫번째 줄은 왼쪽 여백 안쪽으로 '내어밀기'될 것이다.

상자형 문단은 \parindent 값을 0em으로 정의하여 만들 수 있다. 문단과 문단 사이가 식별될 수 있도록 하기 위해서 \parskip을 정수로 설정하는 것이 좋다. 대부분의 타이포그라퍼들은 상자형 문단을 별로 좋아하지 않는데, 이것은 미학적 근거에서만이 아니라 실용적 측면에서도 그러하다. 만약 이전 상자형 문단의 마지막 줄이 줄끝까지 채워지고 그 페이지의 끝에 오는 경우를 생각해보라. 다음번 상자형 문단은 다음 페이지의 꼭대기에서 시작하지만 들여쓰기가 없기 때문에 새로운 문단으로 시작하는 것인지 어떤지를 알 수가 없게 된다.

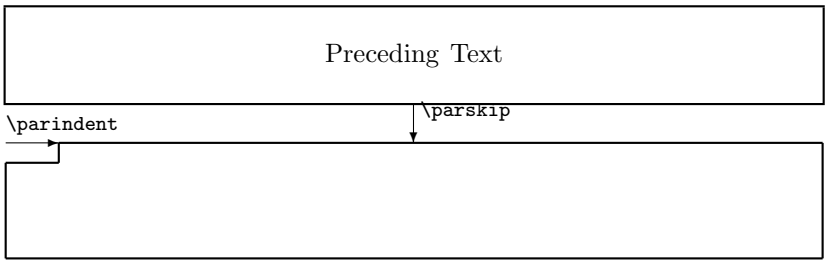


그림 9.1: Paragraphing parameters

LaTeX은 한 문단씩 조판한다는 사실을 알아두는 것은 중요하다. 예를 들면, 한 문단에 적용된 \baselineskip은 그 문단 끝에서 효과가 발생한다. 그리고 크기 선언(예를 들면, \large나 \normalsize 또는 \small)으로 한 문단 안에서 적용된 글꼴 크기는 그 문단의 끝에서 효과가 발생한다.

여러줄 들여밀기

여러줄 내어밀기 문단이란, 문단의 처음 몇 줄의 길이가 나머지 행의 길이와 다르게 설정된 문단을 가리킨다. 표준 내어밀기 문단은 처음 한 줄만 길이를 다르게 설정한 특별한 경우에 해당한다고 생각해도 좋다.

\hangpara{⟨*indent*⟩}{⟨*num*⟩}

\hangpara 명령을 문단의 처음에 지정하면 문단은 여러 줄 내어밀기가 된다. ⟨*indent*⟩ 길이가 양수라면 지정된 행의 왼쪽 끝이 들여밀기 될 것이다. 그러나 그 값이 음수라면 주어진 양만큼 오른쪽 끝이 들여밀기될 것이다. ⟨*num*⟩ 숫자, 예컨대 N이 음수라면 문단의 처음 N행이 들여밀기되지만 N이 양수라면 N+1번째 행에서부터 들여밀기가 이루어진다. 이 문단은 \hangpara{3em}{-3}으로 식자되었다. \hangpara 명령과 문단의 첫 단어 사이에는 공백이 없어야 한다.

\begin{hangparas}{⟨*indent*⟩}{⟨*num*⟩} text \end{hangparas}

hangparas 환경은 hangpara 명령과 같으나, 이 환경 안에 오는 모든 문단이 내어밀기된다는 점이 다르다.

여러줄 내어밀기를 구현한 코드는 hanging 패키지 [Wil01f]의 것과 같다. 몇 가지 예들을 [Thi99]에서 찾아볼 수 있을 것이다.

다른 곳에서 지적했듯이, 장절명령은 내부 매크로 `\@hangfrom` 명령을 타이틀 식자에 이용한다.

`\hangfrom{`⟨*text*⟩`}`

간단한 내어밀기 문단 (이 문단과 같이) 지정하려면 `\hangfrom` 매크로를 사용한다. 이 매크로는 ⟨*text*⟩ 박스를 놓은 다음 그 뒤에 오는 내용을 내어밀기 문단으로 만들어준다. 이 문단은 다음과 같이 한 것이다.

```
\hangfrom{간단한 내어밀기 문단 }(이 문단과 ...
```

9.3 flush와 ragged

flushleft 텍스트는 왼쪽 끝을 세로로 가지런히 정렬하고, flushright 텍스트는 오른쪽 끝을 오른쪽 여백에 잇대어 세로로 가지런히 정렬하는 것이다. 그 반대는 raggedleft와 raggedright인데, 각각 왼쪽 끝과 오른쪽 끝이 들쭉날쭉한 것을 가리킨다. LaTeX은 일반적으로 flushleft와 flushright로 조판한다.

`\begin{flushleft}` text `\end{flushleft}`
`\begin{flushright}` text `\end{flushright}`
`\begin{center}` text `\end{center}`

`flushleft` 환경은 flushleft, raggedright로 식자하는 것이다. `flushright`는 raggedleft, flushright로 식자하는 환경이다. `center`는 raggedleft, raggedright로 식자하고 행을 가운데 가져오는 환경이다. 각각 환경의 시작과 끝에 수직 공백을 조금 넣는다.

`\raggedleft \raggedright \centering`

`\raggedleft` 선언은 텍스트를 raggedleft, flushright로 조판하도록 한다. 마찬가지로 `\raggedright` 선언은 flushleft, raggedright로 조판한다. `\centering` 선언은 raggedleft, raggedright로 조판하고 각 행을 중앙정렬한다. 이 선언들은 환경의 경우와는 달리 여분의 수직 간격을 넣지 않는다.

`\raggedyright[`⟨*space*⟩`]` `\ragrparindent`

`\raggedright`를 폭이 좁은 문단에서 쓰면 오른쪽 끝이 너무 ragged해지고 들여밀기가 되지 않는다. `\raggedyright`는 가능한 폭을 좀 더 채우고 문단을 `\ragrparindent`만큼

들여밀기해준다. 이 값은 \parindent와 같은 값으로 설정되어 있다. ⟨*space*⟩ 옵션 인자는 raggedness 값을 조절하기 위해 사용하는데, 기본값은 2em이다. 다음 예를 보자.

```
\raggedyright[0pt]   % typeset flushright
\raggedyright[1fil]  % same as \raggedright
\raggedyright[0.5em] % less ragged than \raggedright
```

LaTeX이 문단 단위로 조판한다는 사실을 상기하자. 그러므로 \centering과 \raggedleft 선언을 같은 문단 안에서 차례로 주면, 전체 문단은 raggedleft와 flushright로 식자되고 \centering 선언은 문장 끝까지 효과가 발생하지 않을 것이다.

9.4 인용(quotations)

LaTeX은 인용문을 조판하기 위한 두 개의 환경을 제공한다.

\begin{quote} text \end{quote}
\begin{quotation} text \end{quotation}

두 환경 모두 flushleft와 flushright가 되지만 본문의 문단폭(textwidth)보다 조금 줄어든다. 두 환경의 유일한 차이는 quote 환경에서는 문단 들여쓰기가 이루어지지 않지만 quotation 환경에서는 보통 문단과 동일한 모양이 된다는 것이다.

9.5 문단폭 바꾸기

quote와 quotation 환경은 모두 일부 텍스트의 문단폭을 바꾼다. 더 정확하게 말하면 임시로 왼쪽 오른쪽 마진값을 같은 값으로 증가시킨다. 일반적으로 말해서 별로 좋은 발상은 아니지만 이따금 문단폭을 바꾸고 싶어지는 때가 있다.

아래 보인 명령과 환경은 chngpage [Wil01b]에 있는 것과 비슷하지만 몇 가지 점에서 차이가 있다.

\begin{adjustwith}{⟨*left*⟩}{⟨*right*⟩} text \end{adjustwith}
\begin{adjustwith*}{⟨*left*⟩}{⟨*right*⟩} text \end{adjustwith*}

adjustwidth 환경은 일시적으로 ⟨*left*⟩ 길이를 왼쪽 마진에 ⟨*right*⟩를 오른쪽 마진에 추가한다. 즉, 이 값이 양수이면 문단폭이 줄어들고 음수이면 증가한다. quotation 환경은 다음과 같이 한 것과 거의 비슷하다.

```
\begin{adjustwidth}{2.5em}{2.5em}
```

별표붙은 환경 adjustwidth*가 필요한 경우는 왼쪽 마진과 오른쪽 마진이 다른 경우에 쓴다. 별표붙은 환경은 페이지 번호를 체크하여 홀수쪽이면 왼쪽(등쪽)과 오른쪽(배쪽) 여백을 ⟨*left*⟩와 ⟨*right*⟩ 값으로 조절하고 페이지가 짝수이면 왼쪽(배쪽)과 오른쪽(등쪽) 여백을 ⟨*right*⟩와 ⟨*left*⟩ 값으로 조절한다.

```
\strictpagechecktrue \strictpagecheckfalse
```

홀/짝수쪽 페이지 번호 체크는 엄격하게 할 수도 있고 (\strictpagechecktrue), 느슨하게 할 수도 있다(\strickpagecheckfalse). 느슨한 페이지 체크로 하여도 대부분의 경우는 잘 들어맞지만 이따금 잘 되지 않으면 strict 체크가 필요할 때도 있을 것이다.

한 가지 예를 들어보면, figure가 textwidth보다 폭이 넓으면 페이지의 오른쪽 여백까지 벗어나게 될 것이다. 안쪽(등쪽)보다 여백이 넉넉한 바깥쪽(배쪽)으로 튀어나오도록 넓은 그림을 배치하고 싶은 경우, 다음과 같이 해볼 수 있다.

```
\begin{figure}
\centering
\strictpagechecktrue
\begin{adjustwidth*}{0em}{-3em}
% the illustration
\caption{...}
\end{adjustwidth*}
\end{figure}
```

이 예가 이 사용안내서의 188 쪽의 표 [11.4]에 나와 있다. 실제로 판면보다 더 폭이 넓은 표를 넣은 것인데, 이 경우 표를 중간에 오게 하기 위해서 adjustwidth를 이용하여 양쪽 여백 값을 똑같이 줄여주었다.

```
\begin{table}
\begin{adjustwidth}{-1cm}{-1cm}
\centering
...
\end{adjustwidth}
\end{table}
```

adjustwidth 환경은 전체 문단에 적용된다는 사실을 주의하여야 한다. 여러줄 내어밀기 문단이나 좀더 기교적인 \parshape를 사용하지 않는 한 한 문단의 일부에 대해서만 width를 바꿀 수는 없다. 그리고, 여백이 변경된 문단은 페이지를 넘어가더라도 여백값 변경은 여전히 유효하다. 즉, 첫번째 페이지에서 오른쪽을 넓게 잡은 문단이라면 다음 페이지에 넘어가도 오른쪽이 넓은 문단으로 식자된다.

center 환경은 판면을 기준으로 내용을 가운데 정렬한다. 이따금 판면의 중앙이 아니라 페이지를 기준으로 가운데 정렬하고 싶은 경우가 있을 것이다. 예를 들면 판권지(간기)를 식자할 때는 판면이 보이지 않으므로 중앙정렬하면 결과가 한 쪽으로 치우쳐 보일 수 있다.

등쪽에서 배쪽 마진에 가할 변경값을 계산하는 것은 간단하다. §6.4에서 사용했던 기호를 그대로 써서, P_w와 B_w가 각각 재단된 판형와 판면의 폭을 나타낸다고 하고, S와 E가 각각 등쪽 여백과 배쪽 여백을 의미한다고 하면, 다음 M만큼의 값을 등쪽 마진에 더하고 배쪽 마진에서는 빼주면 될 것이다.

$$M = 1/2(P_w - B_w) - S$$

예를 들어보자. \textwidth가 5인치이고, \spinemargin이 1인치라고 가정하자. US letterpaper(\paperwidth는 8.5인치)에서 배쪽 마진이 2.5인치라면 0.75인치[1] 만큼을 등쪽 마진에 더하고 배쪽 마진에서 빼주어야 판면이 중간에 오게 된다. adjustwidth 환경이 이러한 일시적인 변경에 쓰일 수 있다.

```
\begin{adjustwidth*}{0.75in}{-0.75in} ...
```

\calccentering{⟨*length*⟩}

직접 계산하는 것이 귀찮으면 \calccentering을 사용하면 된다. ⟨*length*⟩ 인자는 미리 정의된 길이 명령의 이름이다. \calccentering이 불리면, ⟨*length*⟩를 등쪽 마진에 더하고, 배쪽 마진에서는 빼서 판면을 페이지 중앙에 가져온다.

```
\calccentering{\mylength}
\begin{adjustwidth*}{\mylength}{-\mylength}
text horizontally centered on the physical page
\end{adjustwidth*}
```

\calccentering을 위해서 새로운 길이 변수를 정의할 필요는 없다. 이미 있는 어떤 길이라도 상관없다. 계산이 수행되고 여백을 변경하는 동안 다른 용도로 사용되지 않는 것이기만 하면, \unitlength와 같은 것도 사용할 수 있다. 그리고 필요하다면 기정값을 바꾸어도 된다. \unitlength의 초기값은 1pt이다.

[1] A4 페이퍼에서는 결과가 다를 것이다

9.6 나열 문단(lists)

표준 LaTeX은 네 종류의 리스트를 제공한다. 일반 list 환경은 다른 종류의 리스트를 정의하기 위해 사용되며, description, itemize, enumerate 리스트들이 있다. 이들은 모두 내부적으로 list 환경을 이용하여 정의되어 있다.

이 클래스는 description 리스트는 그대로 제공한다. 그러나 itemize와 enumerate 리스트는 확장하고 있다.

```
\begin{description} \item[⟨label⟩] ... \end{description}
\descriptionlabel⟨style⟩
```

description 리스트는 ⟨*label*⟩ 스타일이 \descriptionlabel 명령의 ⟨*style*⟩ 인자로 주어진다. 기본 정의는 다음과 같다.

```
\newcommand*{\descriptionlabel}[1]{\hspace\labelsep
                                   \normalfont\bfseries #1}
```

이 정의는 레이블을 굵은 글씨로 하는 것이다. 예를 들어 sans 서체 레이블로 바꾸려면 다음과 같이 한다.

```
\renewcommand*{\descriptionlabel}[1]{\hspace\labelsep
                                     \normalfont\sffamily #1}
```

아래 itemize와 enumerate 환경은 enumerate 패키지 [Car98c]에 기초한 것이다.

```
\begin{itemize}[⟨mark⟩] \item ... \end{itemize}
```

itemize에서 \item의 글머리표는 다음과 같다.

1. 불릿 (•\textbullet)
2. 두꺼운 en-대시 (– \bfseries\textendash)
3. 가운데별표 (∗\textasteriskcentered, 및
4. 가운데점 (·\textperiodcentered).

⟨*mark*⟩ 인자는 특정 리스트에서 리스트 항목을 표시하는 글머리표로 쓰인다. 어떤 이유에선가 항목 머리표로 pilcrow 기호를 쓰고자 한다면, 다음과 같이 한다.

```
\begin{itemize}[\P]
\item ...
...
```

`\begin{enumerate}[⟨style⟩] \item ... \end{enumerate}`

`enumerate` 리스트의 세번 째 아이템의 항목 표지는 차례로 3., c., iii., C이다. ⟨*style*⟩ 선택 인자는 항목 번호를 식자하는 형식을 지정하는 데 사용된다. `A`, `a`, `I`, `i`, `1` 가운데 하나를 써서 ⟨*style*⟩을 지정한다. 각각의 카운터는 영문대문자, 영문소문자, 로마숫자 대문자, 로마숫자 소문자, 아라비아 숫자를 나타낸다. 이 글자들은 LaTeX 명령이나 문자로 둘러쌀 수 있지만 그럴 경우 이 특별한 문자들을 중괄호로 감싸준다. 예를 들면 `{a}`와 같이 하라는 것이다. 스타일 지정 문자인지 일반적인 문자인지 구별할 수 있도록 하려는 것이다. 예를 들면, 숫자를 로마숫자 소문자로 하고 여기에 닫는 괄호를 치고 싶은 경우라면 다음과 같이 한다.

```
\begin{enumerate}[i)]
...
```

`\tightlists \defaultlists`

일반적인 LaTeX의 `description`. `itemize`, `enumerate` 리스트들은 각 항목 사이에 수직 간격을 넣어서 넓게 보이도록 조판된다. `\tightlists` 선언이 불린 후에는 이 항목 간 여분의 수직간격이 제거된다. 원래 대로 넓어 보이는 모양으로 식자하고 싶으면 `\defaultlists` 선언을 준다. 이 선언은 해당 리스트 환경 0이전에 두어야 한다.

이 클래스의 기본값은 `\defaultlists`이다. 이 장의 리스트 모양이 앞 장의 모양과 다르게 보이는 것을 알아차렸다면(또는 의식하지 못했더라도), 이 장 시작 부분에 `\tightlists` 선언을 두었기 때문이다.

`\firmlist \tightlist`

`\firmlist` 또는 `\tightlist` 명령은 리스트 환경을 시작한 직후에 사용할 수 있다. 해당 리스트에서 수직 간격을 줄여주는 역할을 한다. `\tightlist`는 모든 수직 간격을 없애고 `\firmlist`는 간격을 갖기는 하지만 보통 리스트와는 조금 다른 스타일로 조판되도록 한다.

`\begin{list}{`⟨*default-label*⟩`}{`⟨*code*⟩`}` items `\end{list}`

LaTeX의 리스트 환경은 `list` 환경을 이용하여 정의된다. 그밖의 몇 가지 환경들, 예를 들면 `quote`, `quotation`, `adjustwidth` 등도 `list`를 이용하여 만들어진 것이다. 그림 (9.2)에는 `list` 환경의 레이아웃을 조절하는 인자들이 나타나 있다.

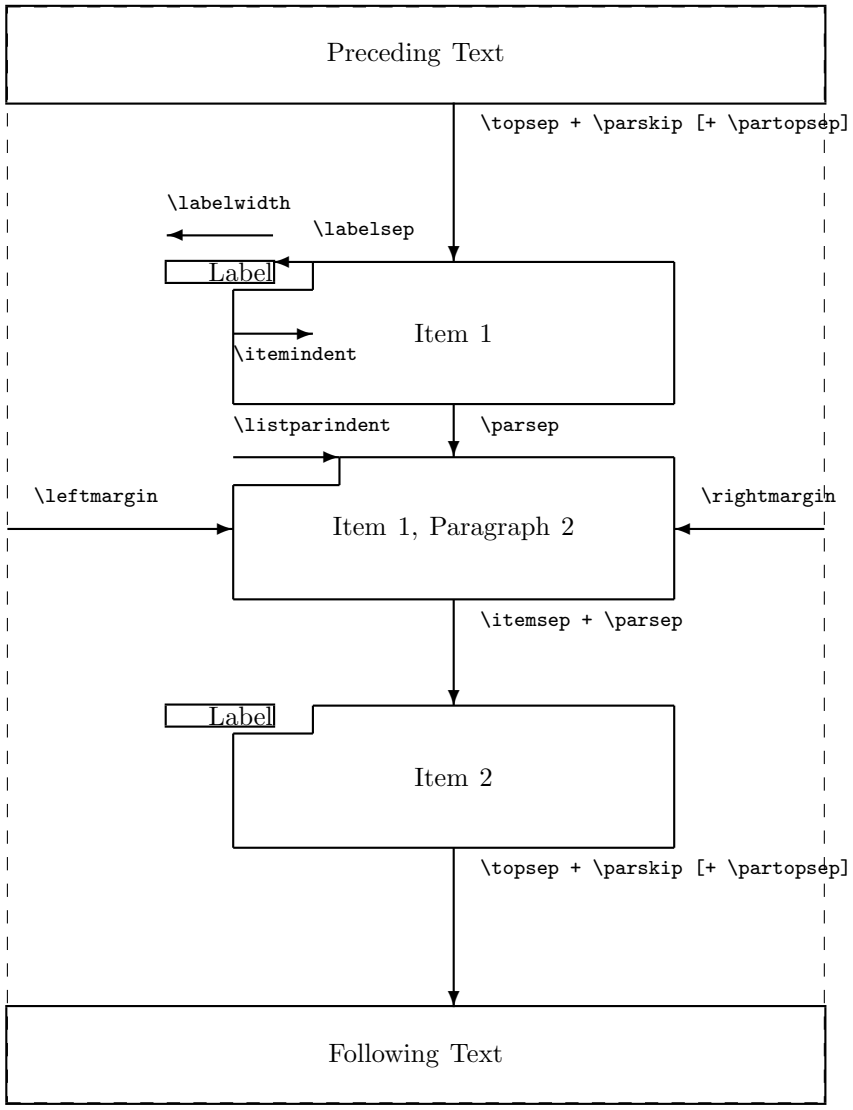


그림 9.2: 일반적인 리스트의 레이아웃 파라미터

list 환경은 두 개의 인자를 갖는다. ⟨*default-label*⟩ 인자는 \item 매크로가 ⟨*label*⟩ 선택인자 없이 사용되었을 때 기본값으로 쓰인다. enumerate의 경우 이 값은 특별히 정의되어 있지만, adjustwidth 등과 같은 대부분 경우 비어 있다.

⟨*code*⟩ 인자는 리스트의 레이아웃 매개변수의 특정 값을 설정하기 위해서 사용한다. 자신만의 리스트를 정의한다면 기본값이 자신의 목적에 알맞지 않을 경우 각각의 파라미터들을 개별적으로 정의하면 된다. 이 파라미터들은 모두 \setlength나 \addtolength 명령으로 재정의할 수 있다.

예를 하나 들어보자. 다음 보기는 각 항목에 bold 글꼴 대신 이탤릭 글꼴을 사용하도록 하고 일반적인 description 리스트보다는 조금 항목간 간격을 좁힌 description 류 리스트이다.

```
%%%%% An italic and tighter description environment
\newcommand{\itlabel}[1]{\hspace\labelsep\normalfont\itshape #1}
\newenvironment{itdesc}{%
  \list{}{%
    \setlength{\labelsep}{0.5em}
    \setlength{\itemindent}{0pt}
    \setlength{\leftmargin}{\parindent}
    \setlength{\labelwidth}{\leftmargin}
    \addtolength{\labelwidth}{-\labelsep}
    \setlength{\listparindent}{\parindent}
    \setlength{\parsep}{\parskip}
    \setlength{\itemsep}{0.5\onelineskip}
    \let\makelabel\itlabel}}{\endlist}
```

위와 같이 정의한 다음에는 보통 리스트를 쓰는 것과 똑같이 사용하면 된다.

```
\begin{itdesc}
\item[label] ....
\end{itdesc}
```

다른 보기를 하나 더 들어보자. 이것은 aglossary라는 이름을 가진 것으로 용어집 (glossary) 과 같은 용도에 사용할 수 있을 것이다.

```
% Glossary list
\newenvironment{aglossary}%
               {\begin{list}{}% empty label
                            {\setlength{\topsep}{\baselineskip}
                             \setlength{\partopsep}{0pt}
                             \setlength{\itemsep}{0.5\baselineskip}
                             \setlength{\parsep}{0pt}
                             \setlength{\leftmargin}{2em}
```

```
                              \setlength{\rightmargin}{0em}
                              \setlength{\listparindent}{1em}
                              \setlength{\itemindent}{0em}
                              \setlength{\labelwidth}{0em}
                              \setlength{\labelsep}{2em}}}%
                {\end{list}}
\newcommand{\gloss}[1]{\item[#1]\mbox{}\\\nopagebreak}
```

다음과 같이 사용한다.

```
\begin{aglossary}
\gloss{TERM 1} definition
\gloss{TERM 2} ...
\end{aglossary}
```

\zerotrivseps \restoretrivseps \savetrivseps

center와 같은 많은 환경들은 trivlist (단순 리스트 형식) 라는 환경을 이용하여 정의된다. 이 환경의 전후에는 약간의 수직간격이 붙는다. 이 간격을 없애려면, \zerotrivseps를 선언하라. 다음과 같은 것을 생각해볼 수 있을 것이다.

```
\newcommand*{\zerotrivseps}{%
  \setlength{\topsep}{0pt}%
  \setlength{\partopsep}{0pt}}
```

이 수직 간격을 다시 회복하려면 \restoretrivseps 선언을 준다. \savetrivseps 명령은 \topsep과 \partopsep 값을 저장하며, \restoretrivseps에 의하여 저장된 값으로 복구된다. 이 클래스 자신이 \savetrivseps를 호출하여 기본값을 기억하도록 되어 있다.

10 목차와 문헌목록, 찾아보기

이 장은 *section* chapterstyle을 사용한다.

10.1 서론

아래 논의는 문서의 시작 부분과 마지막 부분에 오는 요소들에 대해 중점적으로 다루고 있다. 시작 부분에는 목차가 오고 끝에는 찾아보기가 온다. 여기서 구현한 기능은 tocloft와 tocbibind 패키지 [Wil01i, Wil01h]의 것을 합친 것과 같다.

표준 클래스에서 목차(ToC), 그림 목차(LoF) 및 표 목차(LoT)의 디자인은 바꿀 수 없도록 되어 있다. 좀더 정확하게 말하면, 클래스 자체의 정의로 심어져 있다.

10.2 LaTeX이 ToC를 만드는 방법

여기서는 LaTeX이 목차 (ToC)를 어떻게 처리하는지에 대해 개괄적인 설명을 하려 한다. 표목차나 그림목차는 ToC의 경우와 거의 같다.

먼저, 각각의 장절 구분은 101 쪽의 표 [8.1]에 열거된 장절 레벨이 주어져 있다. LaTeX은 `tocdepth` 카운터 값이 엔트리 레벨과 같거나 큰 경우에만 ToC에 해당 장절 엔트리를 식자한다.

```
\maxtocdepth{⟨secname⟩}
\settocdepth{⟨secname⟩}
```

memoir의 `\maxtocdepth` 명령은 `tocdepth` 카운터가 가질 수 있는 최대값을 설정한다. 이 명령은 `\tableofcontents` 명령보다 앞에 두어야 한다. 클래스 기본값은 `\maxtocdepth{section}`으로 설정되어 있다.

memoir 클래스 명령 `\settocdepth`는 §8.4에서 설명한 `\setsecnumdepth`와 비슷한 면이 있다. 이 명령은 `tocdepth` 카운터 값을 설정한 다음 그것을 ToC에 (임시로) 집어넣어서 나열되는 항목 수준을 바꿀 수 있다. 이 명령은 preamble 이후에 쓰여야 하지만

\tableofcontents 명령보다 앞에 불릴 수는 있다. \settocdepth와 \maxtocdepth 매크로는 tocvsec2 패키지 [Wil99b]에서 가져온 것이다.

\addcontentsline{⟨*file*⟩}{⟨*kind*⟩}{⟨*text*⟩}

LaTeX은 문서가 \tableofcontents 명령을 포함하고 있다면 .toc 파일을 만든다. 장절 명령[1]은 LaTeX 명령 \addcontentsline을 불러서 .toc 파일 안에 엔트리를 적어넣는다. 여기서 ⟨*file*⟩은 파일 확장명(예를 들면 toc)이고, ⟨*text*⟩는 (번호붙은) 표제 텍스트이다. 번호가 붙는 경우라면 ⟨*title*⟩ 인자는 {\numberline{number} title-text} 의 형식으로 주어진다.

\phantomsection

NOTE: hyperref 패키지 [Rahtz02]는 \addcontentsline을 사용하는 것을 반기지 않는 것 같다. 이 명령이 hyperref에서 잘 작동하게 하려면 \phantomsection (이 클래스와 hyperref 패키지에 정의되어 있음) 을 \addcontentsline 직전에 두어야 하는 경우가 있다.

\contentsline{⟨*kind*⟩}{⟨*text*⟩}{⟨*page*⟩}

\addcontentsline 명령은 주어진 파일에 엔트리를 다음과 같은 형식으로 쓴다.
\contentsline{⟨*kind*⟩}{⟨*text*⟩}{⟨*page*⟩}
여기서 ⟨*page*⟩는 페이지 번호이다. 각각의 ⟨*kind*⟩에 대해서 LaTeX은 한 가지 명령을 제공하는데,
\l@kind{⟨*title*⟩}{⟨*page*⟩}
이 명령은 \contentsline 엔트리를 실제로 식자한다.

\@pnumwidth{⟨*length*⟩}
\@tocrmarg{⟨*length*⟩}
\@dotsep{⟨*number*⟩}

엔트리를 조판하는 일반적인 레이아웃을 그림 (10.1)에 보였다. 조판 과정에 사용되는 내부 LaTeX 명령은 세 가지가 있다. 페이지 번호는 \@pnumwidth 만큼의 width를 가지는 박스로 flushright 조판된다. 그리고 이 박스는 여백에 잇닿은 오른쪽 끝에 위치한다. 만약 페이지 번호가 너무 길어서 이 박스에 들어가지 못하면 오른쪽 여백 쪽으로 밀려나갈 것이다. 표제 텍스트는 오른쪽 마진에서부터 \@tocrmarg로 주어지는 크기만큼 간격을 주고 식자된다. \@tocrmarg는 \@pnumwidth보다 커야 한다는 점을 기억하자. 일부 엔트리는

[1] 그림과 표의 경우에는 \caption 명령이 .lof와 .lot 파일을 만든다

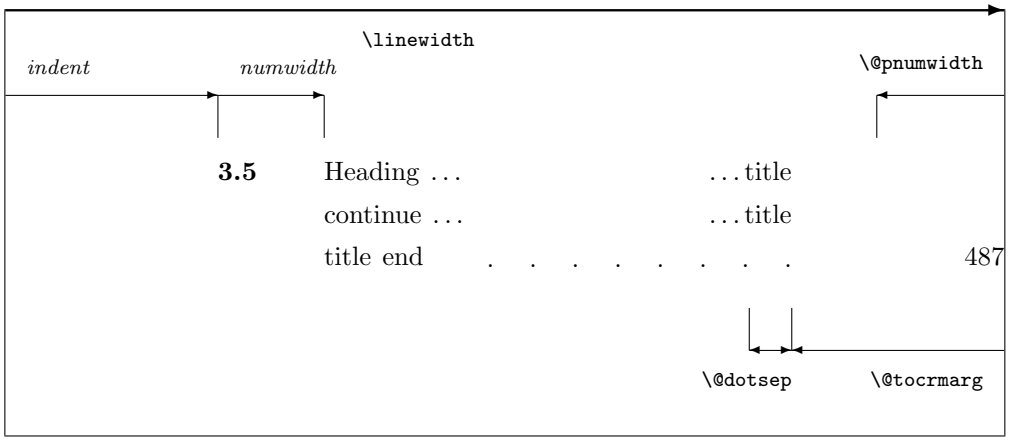


그림 10.1: Layout of a ToC (LoF, LoT) entry

표제의 끝에서 오른쪽 마진까지 점선을 두는 경우도 있다. 점과 점 사이의 간격은 *math units*[2] 단위로 `\@dotsep`에 의하여 지정된다. 표준 클래스에서는 ToC, LoF, LoT에 같은 값이 적용된다.

표준 클래스의 이 값들은 다음과 같다.

- `\@pnumwidth` = 1.55em
- `\@tocrmarg` = 2.55em
- `\@dotsep` = 4.5

각각의 값은, 처음 두 개는 길이변수로 보이지만 `\renewcommand`로 변경할 수 있다.

Part, Chapter의 ToC엔트리는 점선 리더를 식자하지 않는다.

`\numberline{`⟨*number*⟩`}`

`\l@kind` 매크로는 각각 왼쪽 마진으로부터 들여쓰기 값과 *numwidth*를 설정하는 데 쓰인다. `\numberline` 매크로는 숫자를 *numwidth* 폭의 박스로 flushleft 식자한다. 숫자가 너무 길어서 박스에 들어가지 못하면 표제 텍스트와 겹쳐찍힐 것이다. 표제 텍스트는 왼쪽 마진에서 (*indent* + *numwidth*) 만큼 들여밀기된다. 즉, 표제 텍스트는
(`\linewidth` - *indent* - *numwidth* - `\@tocrmarg`) 만큼의 폭을 갖는 상자형 문단으로 식자된다.

표 [10.1]에는 *indent*와 *numwidth* 값이 정리되어 있다. part에 해당하는 *numwidth*는 명시적인 값이 없는 대신, 숫자와 표제 텍스트 사이에 1em의 간격을 준다. 장절명령에

[2] 18mu에서 1em까지

표 10.1: 들어쓰기와 numwidth (단위 em)

Entry	Level	Standard		memoir class	
		indent	numwidth	indent	numwidth
part	-1	0	—	0	—
chapter	0	0	1.5	0	1.5
section	1	1.5	2.3	1.5	2.3
subsection	2	3.8	3.2	3.8	3.2
subsubsection	3	7.0	4.1	7.0	4.1
paragraph	4	10.0	5.0	10.0	5.0
subparagraph	5	12.0	6.0	12.0	6.0
figure/table	(1)	1.5	2.3	0	1.5
subfigure/table	(2)	—	—	1.5	2.3

서 이 값들은 문서 클래스가 `\chapter` 명령을 제공하는가의 여부에 달려 있다. 또한, 좀 이상한 일이기는 하지만, 표와 그림 엔트리들은 모두 들여밀기가 된다.

`\@dottedtocline{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}`

대부분의 `\l@kind` 명령들은 `\@dottedtocline` 명령을 이용하여 정의되어 있다. 이 명령은 세 개의 인자를 가지는데, ⟨*level*⟩ 인자는 표 [10.1]에 보인 레벨을 의미하고 ⟨*indent*⟩와 ⟨*numwidth*⟩는 그림 (10.1)에 그려져 있는 *indent*와 *numwidth*를 가리킨다. 예를 들면, `\l@section` 명령은 다음과 같이 정의된다.

```
\newcommand*{\l@section}{\@dottedtocline{1}{1.5em}{2.3em}}
```

엔트리의 기본 모양을 변경하려면, 이 정의들을 바꾸면 될 것이다. 그러나 이 클래스는 LaTeX 내부명령에 대해서 알지 못하여도 쉽게 이 값을 제어하는 방법을 제공한다.

`\addcontentsline`은 `\contentsline` 명령을 파일에 추가하도록 하는 명령이다.

`\addtocontents{`⟨*file*⟩`}{`⟨*text*⟩`}`

LaTeX은 `\addtocontents` 명령도 제공하는데, 이것은 ⟨*file*⟩에 ⟨*text*⟩를 써넣는 역할을 한다. 추가적인 매크로나 텍스트를 `\tableofcontents`나 그밖의 `listoftables`이 만드는 `.lot` 파일에 써두었다가 나중에 LaTeX이 실행될 때 처리하게 할 수 있다.

`\addcontentsline`과 `\addtocontents`는 인자를 파일로 써주지만 풀리는 명령이 인자로 오게될 때는 `\protect`해주어야 한다.

ToC 등의 레이아웃을 변경하려 할 때 사용할 수 있다. 예를 들어보면,

- 만약 페이지 번호가 오른쪽 여백을 먹어들어가면,

```
\renewcommand{\@pnumwidth}{3em}
```

```
\renewcommand{\@tocrmarg}{4em}
```

그렇지만 길이값은 주의깊게 사용해야 한다.

- ToC등의 (장절) 표제를 하이픈없이 raggedright로 조판하려면,

  ```
  \renewcommand{\@tocrmarg}{2.55em plus1fil}
  ```

  여기서 2.55em 값은 원하는 여백값으로 바꿀 수 있다.
- 리더의 점 사이의 길이 \@dotsep 값을 아주 크게 만들어서 사라지게 할 수 있다.

  ```
  \renewcommand{\@dotsep}{10000}
  ```

  점선 리더를 ToC와 LoF에는 붙이고 LoT에서는 빼고 싶을 때.

  ```
  ...
  \tableofcontents
  \makeatletter \renewcommand{\@dotsep}{10000} \makeatother
  \listoftables
  \makeatletter \renewcommand{\@dotsep}{4.5} \makeatother
  \listoffigures
  ...
  ```
- Part 엔트리 아래에 선을 긋고 싶을 때

  ```
  \part{Part title}
  \addtocontents{toc}{\protect\mbox{}\protect\hrulefill\par}
  ```

  이미 언급한 대로, \addtocontents와 \addcontentsline에는 풀리는 명령이 올 때 protect해주어야 한다. 위의 예의 결과는 .toc 파일에 다음 두 개의 선이 따라올 것이다. (제2절이 34페이지에 있을 경우)

  ```
  \contentsline {part}{II\hspace {1em}Part title}{34}
  \mbox {}\hrulefill \par
  ```

  만약 \protect가 사용되지 않으면, 두번째 선은 다음과 같이 풀려버린다.

  ```
  \unhbox \voidb@x \hbox {}\unhbox \voidb@x \leaders \hrule \hfill \kern \z@ \par
  ```
- ToC에 찍히는 엔트리의 레벨을 바꾸려면(예를 들어 ToC에는 일반적인 subsection을 나타내고 부록에는 메인 타이틀만 적고 싶을 때),

  ```
  \appendix
  \settocdepth{chapter}
  \chapter{First appendix}
  ...
  ```

@ 기호가 포함된 명령을 수정하려 할 때는 .sty 파일 안에서만 가능하다. 만약 문서에서 이것을 수정하려면 \makeatletter와 \makeatother로 감싸주어야 한다는 사실을 기억하자.

```
\makeatletter
\renewcommand{\@dotsep}{9.0}
\makeatother
```

10.3 이 클래스가 ToC를 만드는 방법

이 클래스는 ToC 등의 모양을 변경하는 여러 가지 방법을 제공한다. 위에 설명한 것에 대해 굳이 염려할 필요가 없다.

```
\tableofcontents \tableofcontents*
\listoffigures \listoffigures*
\listoftables \listoftables*
```

ToC, LoF, LoT 들은 이 명령이 선언된 위치에서 식자된다. 그러나 표준 LaTeX클래스와 이 클래스는 두 가지 차이점이 있다. 표준 LaTeX 클래스에서는 `\chapter` 표제와 ToC, LoF, LoT가 새로운 페이지로 시작한다. 이 클래스에서는 반드시 새로운 페이지로 시작할 필요가 없다. 새 페이지로 시작하게 하고 싶으면 적당한 명령을 앞에서 불러주면 된다. 예를 들어보자.

```
 ...
\clearpage
\tableofcontents
\clearpage
\listoftables
...
```

그리고, 별표붙지 않은 명령은 자신의 헤딩을 ToC에 추가한다. 그렇게 하고 싶지 않으면 별표붙은 명령을 쓰면 된다.

표제 바꾸기

ToC, LoF, LoT의 표제 모양을 제어하는 명령이 제공된다.

```
\contentsname \listfigurename \listtablename
```

LaTeX의 관례를 따라, 표제 텍스트는 `\contentsname`, `\listfigurename`, `\listtablename` 명령의 값으로 설정된다.

ToC, LoF, LoT 표제의 식자를 제어하는 명령들은 거의 비슷한 모양을 하고 있으므로, 편의를 위해서 앞으로 `Z`를 'toc', 'lof', 'lot'를 가리키는 표지로 사용하겠다. 즉, `\Zmark`란, `\tocmark`, `\lofmark`, `\lotmark` 등을 가리킨다.

ToC 표제를 식자하는 코드는 다음과 같다.

```
\tocheadstart
```

```
\printtoctitle{\contentsname}
\tocmark
\thispagestyle{chapter}
\aftertoctitle
```

여기 쓰인 매크로는 아래에서 따로 설명한다.

`\Zheadstart`

이 매크로는 타이틀이 실제로 프린트되기 전에 불려야 한다. 기본값 정의는 아래와 같다.

```
\newcommand{\Zheadstart}{\chapterheadstart}
```

`\printZtitle{`⟨*title*⟩`}`

표제는 printZtitle을 통해서 식자된다. 기본적으로 \printchaptertitle을 실제 식자에 이용하도록 설정되어 있다.

`\Zmark`

이 매크로들은 ToC, LoF, LoT 페이지 면주의 running heads 모양을 결정한다. 기본값은 다음과 같다.

```
\newcommmand{\Zmark}{\@mkboth{\...name}{\...name}}
```

\...name은 \contentsname이거나 \listfigurename, \listtablename이다. 이것을 바꿀 필요는 없겠지만, 특정 \pagestyle에서는 다르게 변경하는 경우도 있을 수 있다.

`\afterZtitle`

이 매크로는 표제가 식자된 후에 불린다. 기본적으로 \afterchaptertitle로 설정되어 있다.

본질적으로 ToC, LoF, LoT 표제는 chapter 표제와 동일한 형식을 갖는다. 따라서 현재의 chapterstyle에 의하여 모양이 결정될 것이다. 이 모양을 바꾸려면 별도의 chapterstyle을 쓰거나 몇 가지 매크로를 바꾸어주면 된다. 다음에 예가 나와 있다.

- 다음과 같이 한다.

```
\renewcommand{\printZtitle}[1]{\hfill\Large\itshape #1}
```

그러면 표제가 오른쪽 정렬되어 Large italic 폰트로 식자된다.

- 만약 Large bold 중앙정렬된 표제가 필요하다면,

```
\renewcommand{\printZtitle}[1]{\centering\Large\bfseries #1}
```

위와 같이 한다.

- 다음과 같이 쓰면,

```
\renewcommand{\afterZtitle}{\thispagestyle{empty}\afterchaptertitle}
```

결과는 목차의 첫 페이지는 *empty* 페이지 스타일이 될 것이다. 기본값은 *chapter* 페이지스타일을 쓰도록 되어 있다.

- 다음은,

```
\renewcommand{\afterZtitle}{%
  \par\nobreak \mbox{}\hfill{\normalfont Page}\par\nobreak}
```

'Page'라는 단어를 표제가 앉혀진 줄에 flushright로 찍힌다.

엔트리 식자

여러 가지 엔트리들의 식자를 더 잘 통제할 수 있도록 하는 명령들도 제공된다. 기본적인 엔트리 레이아웃에 대해서는 layouts 패키지 [Wil99a] 또는 [GMS94, 34쪽]에 잘 나와 있는데, 그림 (10.1)에 그것을 옮겨두었다.

`\cftdot{`⟨*text*⟩`}`

ToC 기본값은 장절명령 가운데서 비교적 하위단위에 대해서만 표제와 페이지 번호 사이에 점선 리더(안내선)을 가지도록 되어 있다. 이 클래스는 모든 엔트리에 대해서 안내선을 그을 수 있게 한다. 안내선 안의 '점'은 `\cftdot`의 값이다. 기본 정의는 `\newcommand{\cftdot}{.}` 인데, 이것이 기본 점선 안내선을 만든다. `\cftdot`을 재정의하여 마침표로 이루어지는 점선 대신 어떤 기호라도 사용할 수 있다. 예를 들면,

```
\renewcommand{\cftdot}{\ensuremath{\ast}}
```

위와 같이 하면 점선 안내선가 별표 안내선으로 바뀐다.

`\cftdotsep`
`\cftnodots`

각 엔트리 유형들은 안내선의 점 사이 간격을 조절할 수 있다. 일관성을 유지하기 위해서 모든 점선 안내선에 동일한 간격을 주는 것이 좋다. \cftdotsep 매크로는 기본 간격을 지정한다. 그러나, 만약 간격이 아주 넓어지면 점이 나타나지 않는다. \cftnodots는 이 점 사이 간격을 '아주 크게' 만드는 것이다.

\setpnumwidth{⟨*length*⟩}
\setrmarg{⟨*length*⟩}

페이지 번호는 고정폭 박스에 식자된다. \setpnumwidth 명령은 이 박스의 폭(LaTeX 내부 명령은 \@pnumwidth)을 변경한다. 표제 텍스트는 오른쪽 마진 위치 직전까지 이어진다. \setrmarg 명령은 이 거리(LaTeX 내부명령은 \@tocrmarg)를 조절한다. \setrmarg에 사용된 길이는 반드시 \setpnumwidth보다 커야 한다는 것을 알아두자. 이 값은 문서에서 항상 일정하게 유지된다.

이 안내서는 기본값보다 페이지 번호에 좀더 넉넉한 값을 주어야 했다. 그래서 다음과 같이 정의하였다.

```
\setpnumwidth{2.55em}
\setrmarg{3.55em}
```

\cftparskip

일반적으로 ToC 등의 \parskip은 0이다. 이 값은 \cftparskip 길이를 변경하여 바꿀 수 있다. 현재의 \cftparskip 값이 ToC, LoF, LoT에 사용되지만 \tableofcontents나 \listoffigures, \listoftables 명령을 부르기 전에 이 값을 변경하여 (별로 좋은 생각은 아니지만) 각각의 목차마다 서로 다른 값을 갖도록 할 수도 있다.

이 이후로 X라고 표시하는 것은 다음과 같은 것을 가리키는 것으로 한다.

- part는 \part 표제.
- chapter는 \chapter 표제.
- section는 \section 표제.
- subsection는 \subsection 표제.
- subsubsection는 \subsubsection 표제.
- paragraph는 \paragraph 표제.
- subparagraph는 \subparagraph 표제.
- figure는 그림 \caption 표제.
- subfigure는 subfigure \caption 표제.
- table은 표 \caption 표제.

- subtable은 subtable \caption 표제.

\cftchapterbreak

\l@chapter가 \chapter 엔트리를 ToC에 식자할 때 맨 먼저 하는 일은 \cftchapterbreak를 부르는 것이다. 이 매크로는 다음과 같이 정의되어 있다.

```
\newcommand{\cftchapterbreak}{\addpenalty{-\@highpenalty}}
```

이것은 되도록 엔트리의 뒤가 아니라 앞에서 페이지 나눔이 일어나도록 한다. 만약 원한다면 \cftchapterbreak가 다른 일을 하도록 수정할 수도 있다.

\cftbeforeXskip

이 길이변수는 엔트리 앞의 수직 간격을 제어한다. \setlength로 바꿀 수 있다.

\cftXindent

이 길이변수는 왼쪽 여백에서 엔트리까지의 들여밀기 간격(그림 (10.1)의 *indent*)을 제어한다. \setlength로 바꿀 수 있다.

\cftXnumwidth

이 길이변수는 표제 번호(그림 (10.1)의 *numwidth*)를 식자하기 위해 필요한 간격을 제어한다. \setlength로 바꿀 수 있다. 표제가 길거나 여러 줄일 때, 다음 줄부터는 이 값만큼 들여쓰기 된다.

나머지는 엔트리를 식자하는 데 관련된 명령들이다. 번호가 붙거나 붙지 않았을 때 식자가 어떻게 이루어지는지를 보여주는 간단한 pseudo-code를 보자.

```
{\cftXfont {\cftXpresnum SNUM\cftXaftersnum\hfil} \cftXaftersnumb TITLE}
        {\cftXleader}{\cftXpagefont PAGE}\cftXafterpnum\par

{\cftXfont TITLE}{\cftXleader}{\cftXpagefont PAGE}\cftXafterpnum\par
```

SNUM은 section 번호이다. TITLE은 표제 텍스트이다. PAGE는 페이지 번호이다. 번호붙은 엔트리에서 가상 코드
{\cftXpresnum SNUM\cftaftersnum\hfil}
이것은 \cftXnumwidth 폭을 갖는 박스 안에 놓인다.

\cftXfont

이 매크로는 표제의 모양(만약 번호붙은 표제라면 번호부까지)을 제어한다. \renewcommand로 바꿀 수 있다.

\cftXpresnum \cftXaftersnum \cftXaftersnumb

절번호는 \cftXnumwidth 폭을 갖는 박스에 식자된다. 이 박스 안에서 \cftXpresnum이 가장 먼저 불리며, 그 다음에 숫자가 놓인다. \cftXafternum 매크로는 숫자가 식자된 이후에 온다. 이 명령이 \hfil이면 박스 내용이 flushleft 된다. 박스가 식자된 후에 \cftXaftersnumb 매크로가 표제 텍스트를 식자하기 전에 불린다. 이 세 개의 매크로는 모두 \renewcommand로 바꿀 수 있다. 기본값은 아무 것도 하지 않는 것이다.

\partnumberline{⟨*num*⟩}
\chapternumberline{⟨*num*⟩}

ToC 안에서 \partnumberline과 \chapternumberline 매크로는 각각 \part와 \chapter 숫자를 식자하는 역할을 한다. 내부적으로 위에 언급한 \cftXpresnum, \cftXafternum, \cftaftersnumb를 이용한다. 만약 \chapter의 숫자를 없애고 싶다면, 다음과 같이 하라.

```
\renewcommand{\chapternumberline}[1]{}
```

hyperref [Rahtz02]은 \partnumberline과 \chapternumberline 명령을 이해하지 못하므로, 이 패키지를 사용하려 한다면 memhfixc 패키지를 로드하도록 한다. 이 패키지는 memoir와 함께 제공된다.

\cftXleader
\cftXdotsep

\cftXleader는 표제와 페이지 번호 사이의 안내선을 정의한다. \renewcommand로 바꿀 수 있다. 안내선의 점 사이의 공백은 \cftXdotsep(그림 (10.1)의 \@dotsep)에 의하여 제어된다. \renewcommand로 바꿀 수 있으며 그 값은 숫자일 수도 있고 \cftdotsep이나 \cftnodots(이 때는 점이 나타나지 않는다)일 수도 있다. 이 간격은 18mu에서 1em까지 *math units*로 정의된다.

\cftXpagefont

이것은 페이지 숫자를 식자할 때 사용할 폰트를 지정한다. \renewcommand로 바꾼다.

\cftXafterpnum

이 매크로는 페이지 번호가 식자된 다음에 불린다. 기본값은 아무것도 하지 않는 것이다. \renewcommand로 바꿀 수 있다.

\cftsetindents{⟨*entry*⟩}{⟨*indent*⟩}{⟨*numwidth*⟩}

\cftsetindents 명령은 ⟨*entry*⟩의 *indent*를 ⟨*indent*⟩ 길이로, 그리고 *numwidth*를 ⟨*numwidth*⟩ 길이로 설정한다. ⟨*entry*⟩ 인자는 표준 엔트리(예. subsection) 가운데 하나의 이름이거나, 문서에서 정의된 엔트리 이름이다.

```
\cftsetindents{figure}{0em}{1.5em}
```

위의 예는 그림 엔트리를 왼쪽 정렬시킨 것이다.

이 안내서는 ToC의 절 숫자에 기본값(세 자리)보다 더 많은 공간이 필요했다. 그래서 preamble에 다음과 같이 정의하였다.

```
\cftsetindents{section}{1.5em}{3.0em}
\cftsetindents{subsection}{4.5em}{3.9em}
\cftsetindents{subsubsection}{8.4em}{4.8em}
\cftsetindents{paragraph}{10.7em}{5.7em}
\cftsetindents{subparagraph}{12.7em}{6.7em}
```

한 수준의 indent 값을 바꾸면 그 아래 수준의 indent도 함께 바뀐다는 점을 알아두자.

\cftXfont, \cftXaftersnum, \cftXaftersnumb, \cftXleader, \cftXafterpnum 정의를 하나 이상 바꾸면 여러 가지 효과를 줄 수 있다. 예를 들어보겠다. 먼저 다음과 같은 기본 정의가 주어져 있다 하자.

```
\newcommand{\cftXfont}{}
\newcommand{\cftXaftersnum}{}
\newcommand{\cftXaftersnumb}{}
\newcommand{\cftXleader}{\cftdotfill{\cftXdotsep}}
\newcommand{\cftXdotsep}{\cftdotsep}
\newcommand{\cftXpagefont}{}
\newcommand{\cftXafterpnum}{}
```

(일관성을 유지하기 위해 동일한 글꼴이 표제와 안내선, 그리고 페이지 숫자에 사용되어야 한다는 점을 명심하자.)

- 안내선의 점을 제거한다.

```
\renewcommand{\cftXdotsep}{\cftnodots}
```

- 표제 (숫자) 앞에 무엇인가(이름 등)를 둔다.

```
\renewcommand{\cftXpresnum}{SOMETHING }
```

- 절 숫자 뒤에 콜론을 붙인다.

```
\renewcommand{\cftXaftersnum}{:}
```

- 표제 숫자 앞에 무엇인가를 두고, 표제 숫자 뒤에는 콜론을 붙이고, 전부 볼드체로 하면서 표제 텍스트는 그 다음 줄에서 시작하도록 만든다.

```
\renewcommand{\cftXfont}{\bfseries}
\renewcommand{\cftXleader}{\bfseries\cftdotfill{\cftXdotsep}}
\renewcommand{\cftXpagefont}{\bfseries}
\renewcommand{\cftXpresnum}{SOMETHING }
\renewcommand{\cftXaftersnum}{:}
\renewcommand{\cftXaftersnumb}{\\}
```

숫자 박스에 숫자 이외의 텍스트를 추가하려 한다면, 박스의 폭을 증가시켜야 여러 줄 타이틀이 가지런하게 세로정렬될 것이다. 박스의 폭을 바꾸는 것은 대개 indent 값도 바꾸어야 한다는 것을 의미한다. 위의 예에 해당하는 박스 폭 조절 방법 중의 하나는 다음과 같다.

```
\newlength{\mylen}                  % a "scratch" length
\settowidth{\mylen}{\bfseries\cftXpresnum\cftXaftersnum}
\addtolength{\cftXnumwidth}{\mylen} % add the extra space
```

- 절 숫자를 flushright로 설정한다.

```
\setlength{\mylen}{0.5em}    % extra space at end of number
\renewcommand{\cftXpresnum}{\hfill} % note the double 'l'
\renewcommand{\cftXaftersnum}{\hspace*{\mylen}}
\addtolength{\cftXnumwidth}{\mylen}
```

위의 예에서 추가된 `\hfill`이 박스의 마지막 `\hfil`을 덮어쓴다. 그러므로 전부 박스의 오른쪽 끝으로 이동한다. 추가 공백은 숫자가 타이틀 표제의 왼쪽에 바짝 붙어서 식자되지 않도록 한다.

- 엔트리를 raggedleft로 설정한다. 그러나 이 설정은 모든 표제가 한 줄일 때만 그럭저럭 괜찮아보일 것이다.

```
\renewcommand{\cftXfont}{\hfill\bfseries}
\renewcommand{\cftXleader}{}
```

- 페이지 번호를 오른쪽 끝에 붙이지 않고 엔트리 바로 뒤에 붙여서 식자한다.

```
\renewcommand{\cftXleader}{}
\renewcommand{\cftXafterpnum}{\cftparfillskip}
```

`\cftparfillskip`

`\parfillskip` 값은 한 문단의 마지막 줄을 채우도록 설정하는 값이다. 이것은 `\cftXleader`에 엄청나게 큰 어간 값을 주어서 표제의 마지막 줄을 채우도록 한 것이다. `\cftparfillskip` 명령은 이런 효과를 발생하도록 한다.

`\cftpagenumbersoff{`⟨*entry*⟩`}`
`\cftpagenumberson{`⟨*entry*⟩`}`

`\cftpagenumbersoff` 명령은 ⟨*entry*⟩의 페이지 번호를 나타내지 않는다. ⟨*entry*⟩는 표준적인 엔트리 유형의 이름(e.g., `subsection`, `figure`)이거나, 문서에서 새롭게 정의된 엔트리 이름이다.

`\cftpagenumberson` 명령은 ⟨*entry*⟩에 대하여 `\cftpagenumbersoff`에 상당하는 효과를 낸다.

`comp.text.tex` 뉴스그룹에 올라온 질문 중에, 부록의 타이틀을 페이지 번호 없이 ToC에 올리는 방법이 무엇인지를 물어온 것이 있었다. 다음과 같이 하면 간단하다.

```
...
\appendix
\addtocontents{toc}{\cftpagenumbersoff{chapter}}
\chapter{First appendix}
```

만약 부록 이후에 chapter 유형의 헤딩이 ToC에 들어오는 것(아마도 참고문헌이거나 찾아보기일 것인데)이 있다면, 다음처럼 하는 방법이 있다.

```
\addtocontents{toc}{\cftpagenumberson{chapter}}
```

부록 이후에는 ToC에서의 페이지 번호붙이기를 복구한다.

이 이외에 무엇을 할 수 있을지는 독자의 창조적 능력에 맡겨두기로 하겠다. 그러나, 뭔가 이보다 발전한 결과를 얻었다면 이 문서의 다음번 버전에 포함시킬 수 있도록 필자에게 알려주기 바란다.

10.4 New list of... 와 엔트리

`\newlistof{`⟨*listofcom*⟩`}{`⟨*ext*⟩`}{`⟨*listofname*⟩`}`

`\newlistof` 명령은 새로운 'List of...'를 만든다. 첫번째 인자인 ⟨*listofcom*⟩은 `\listofcon`이라는 이름을 가지는 새로운 명령을 정의한다. 이것은 `\listoffigures`와 같이 사용하여

'List of...'를 식자한다. ⟨*ext*⟩ 인자는 새로운 리스팅에 사용될 파일의 확장명이다. 마지막 인자인 ⟨*listofname*⟩은 'List of...'의 표제이다. `\listofcom`은 별표를 붙이거나 붙이지 않을 수 있다. 별표를 붙이지 않으면 `\listofcom`은 ⟨*listofname*⟩을 ToC에 추가한다. 반면 별표를 붙인 `\listofcom*`은 ToC에 표제를 추가하지 않는다.

예를 들어보자.

```
\newcommand{\listanswername}{List of Answers}
\newlistof{listofanswers}{ans}{\listanswername}
```

이 명령은 새로운 `\listofanswers` 명령을 정의하여 `\listanswername`으로 주어진 표제 아래 answers를 식자한다. 이 때 사용되는 answer 표제는 `.ans` 파일에 들어 있다. 'answer' 코드를 어떻게 정의하느냐는 것은 사용자의 책임이다. 예컨대,

```
\newcounter{answer}[chapter]
\renewcommand{\theanswer}{\thechapter.\arabic{answer}}
\newcommand{\answer}[1]{
  \refstepcounter{answer}
  \par\noindent\textbf{Answer \theanswer. #1}
  \addcontentsline{ans}{answer}{\protect\numberline{\theanswer}#1}\par}
```

여기서 `\answer{Hard} The \ldots`와 같이 쓰면 결과는 다음과 같을 것이다.

Answer 1. Hard
The ...

앞서 말한 대로, `\newlistof` 명령은 `\listofcom` 이외에도 여러 가지 새로운 명령을 추가한다. 그 대부분은 익숙한 것들이다. 편의상 `\newlistof{...}{Z}{...}`이 Z라는 새로운 파일 확장명을 가지는 것으로 하고, Z는 §10.3에서 사용한 것과 같이 쓰기로 하자. 그러면 `\listofcom`은 다음과 같은 새로운 명령을 사용가능하게 한다.

다음 네 개의 명령, 즉, `\Zmark`, `\Zheadstart`, `\printZtitle`, `\afterZtitle`들은 §10.3에서 기술한 것과 같은 이름의 명령들이다. (이 클래스는 내부적으로 ToC, LoF, LoT를 정의하기 위해 `\newlistof` 매크로를 사용하고 있다.) 특히 `\Zmark`의 기본 정의는 다음과 같다.

```
\newcommand{\Zmark}{\@mkboth{listofname}{listofname}}
```

이것은 사용되는 `\pagestyle`이 무엇이냐에 따라 달라질 수 있다.

Zdepth

Zdepth 카운터는 표준 tocdepth 카운터와 유사하다. 새로 정의한 리스팅에서 번호 레벨이 Zdepth보다 크면 식자되지 않는다. 기본 정의는 다음과 같다.

```
\setcounter{Zdepth}{1}
```

\insertchapterspace \addtodef{⟨*macro*⟩}{⟨*prepend*⟩}{⟨*append*⟩}

\chapter 명령은 \insertchapterspace를 사용하여 LoF와 LoT에 수직 간격을 삽입한다는 사실을 상기하자. \insertchapterspace를 수정하여, 이런 여분의 간격을 새로운 리스팅에 추가할 수 있다. 가장 쉬운 방법은 \addtodef 매크로를 이용하는 것이다. 다음과 같이.

```
\addtodef{\insertchapterspaces}{}%
  {\\ref{tab:indents}addtocontents{ans}{\protect\addvspace{10pt}}}
```

\addtodef 매크로는 §17.9에서 소개할 것이다.

새로운 'List of...'를 만드는 데 있어서 또 한가지는, 엔트리의 형식을 지정하는 것이다. 즉, 적절한 \l@kind 매크로를 정의하는 것이다.

\newlistentry[⟨*within*⟩]{⟨*cntr*⟩}{⟨*ext*⟩}{⟨*level-1*⟩}

\newlistentry 명령은 'List of...'에 엔트리를 식자하기 위해 필요한 명령을 정의한다. 첫번째 인자 ⟨*cntr*⟩은 새로운 카운터 cntr가 정의되어 있지 않다면 정의한다. ⟨*within*⟩ 선택 인자는 within이 변경되면 그때마다 cntr 카운터가 1로 재설정되도록 하기 위해 쓰인다. 즉, 첫번째 두 개의 인자는 cntr이 이미 정의되어 있지 않을 때를 위한 것으로, \newcounter{⟨*cntr*⟩}[⟨*within*⟩] 을 부르는 것과 거의 같다. 만약 cntr이 이미 정의되어 있다면 \newcounter 를 부르지 않는다. cntr은 엔트리 표제와 함께 오는 숫자를 식자하는 데 쓰인다.

두번째 인자 ⟨*ext*⟩는 엔트리 리스팅의 파일 확장명이다. 마지막 인자 ⟨*level-1*⟩은 번호 레벨에서 1을 뺀 값을 나타낸다.

\newlistentry는 여러 가지 명령을 정의한다. 만약 \newlistentry{within}{X}{Z}{N} 으로 불려진 경우를 생각해보자. 여기서 X와 Z는 앞서 사용된 것과 같은 뜻이다. 즉 Z는 파일 확장명이다. N이 정수라면, 다음 명령을 사용할 수 있다.

\cftbeforeXskip, \cftXfont, \cftXpresnum, \cftXaftersnum, \cftXaftersnumb, \cftXleader, \cftXdotsep, \cftXpagefont, \cftXafterpnum, 이 명령들은 모두 §10.3에서 보인 이름과 동일한 명령들이다. 기본값은 앞서 이미 소개하였다.

\cftXindent와 \cftXnumwidth의 기본값은 ⟨*level-1*⟩ 인자 값(즉, 예제의 N)에 따라 설정된다. N=0이라면 이 값은 memoir 클래스의 표 [10.1]에 나열된 대로 그림과 표에 적용된다. N=1이면 이 설정은 subfigure 등에 적용된다. N의 값이 0보다 작거나 4보다 크거나 기본값이 아니라면 이 값을 설정하려면 \cftXindents 명령을 사용하라.

\l@X는 엔트리를 목차에 식자하는 내부 명령으로서, 앞서 소개한 \cft*X* 명령을 이용하여 정의되어 있다. \Zdepth가 N이거나 그보다 작으면 엔트리는 식자되지 않는다. Z는 리스트 파일 확장명이다.

\theX 명령은 X 카운터 값을 인쇄한다. 기본값으로는 아라비아 숫자를 표시하도록 되어 있다. ⟨*within*⟩ 인자가 사용되면 \theX는 다음과 같이 정의된다.

```
\renewcommand{\theX}{\thewithin.\arabic{X}}
```

그밖의 경우에는 다음과 같다.

```
\renewcommand{\theX}{\arabic{X}}
```

\newlistentry를 독립적으로 사용하는 예제를 보자. 다음은 sub-answers를 설정하는 것이다.

```
\newlistentry[answer]{subanswer}{1}
\renewcommand{\thesubanswer}{\theanswer.\alph{subanswer}}
\newcommand{\subanswer}[1]{
   \refstepcounter{subanswer}
   \par\textbf{\thesubanswer) #1}
   \addcontentsline{ans}{subanswer{\protect\numberline{\thesubanswer}#1}}
\setcounter{ansdepth}{2}
```

그 다음에,

```
\answer{Harder} The \ldots
  \subanswer{Reformulate the problem} It assists \ldots
```

위의 코드는 다음과 같이 식자한다.

Answer 2. Harder

The ...

2.a) Reformulate the problem It assists ...

answer 엔트리가 List of Answers 목차 (\listofanswers 명령으로 식자됨)에 나타나게 될 것이다. subanswers가 보이게 하기 위해서 \setcounter{ansdepth}{2} 명령이 사용되었다.

subanswers에 페이지 번호 붙이기를 억제하려면 다음과 같이 한다.

```
\cftpagenumbersoff{subanswer}
```

\newlistentry의 또다른 보기로, subparagraph보다 하위 수준의 장절구분이 필요한 경우를 생각해보자. 이것을 subsubpara라고 부르기로 한다. \subsubpara 명령 자체는 LaTeX 핵심명령인 \@startsection으로 정의할 수 있다. 그리고 \subsubparamark 매크로, subsubpara 카운터, \thesubsubpara 매크로, \l@subsubpara 매크로를 정의해야 한다. \newlistentry를 사용하는 예를 다음에 보였으니 살펴보라. @ 기호가 포함된 명령에 주의할 것.

```
\newcommand{\subsubpara}{\@startsection{subpara}
   {6}                            %                    level
   {\parindent}                   %  indent from left margin
   {3.25ex \@plus1ex \@minus .2ex}  %     skip above heading
   {-1em}       runin heading with  % 1em between title & text
   {\normalfont\normalsize\itshape} % italic number and title
}
\newlistentry[subparagraph]{subsubpara}{toc}{5}
\cftsetindents{subsubpara}{14.0em}{7.0em}
\newcommand*{\subsubparamark}[1]{}  % gobble heading mark
```

'List of...'는 리스트 엔트리를 저장하기 위한 파일을 각각 사용한다. 이 파일들은 문서가 처리되는 동안 파일쓰기를 위해 열려 있어야 한다. TeX은 한번에 열 수 있는 파일 수에 제한이 있고, 이 때문에 사용할 수 있는 리스트의 수도 한계가 있다. 찾아보기나 참고문헌을 전혀 사용하지 않고 오로지 ToC들만 가진 문서에서 LoX, 즉 LoF와 LoT등의 최대 수는 11을 넘지 못한다. 너무 많은 새로운 목차를 정의하면 LaTeX은 다음과 같은 에러 메시지를 보여준다.

```
No room for a new write
```

이 에러를 만났을 때 할 수 있는 유일한 일은 문서를 다시 설계하는 것이다.

예제 — plate

앞서 언급한 바 있는 대로, 단행본 책에는 삽화 부분이 들어가는 경우가 있다. 이 부분만 광택지를 사용하고 나머지 부분은 보통용지를 사용할 때 이 부분을 'plate'라고 부르는 경우가 많다. plate에 해당하는 새로운 목차를 정의해보자.

```
\newcommand{\listplatename}{Plates}
\newlistof{listofplates}{lop}{\listplatename}
\newlistentry{plate}{lop}{0}
\cftpagenumbersoff{plate}
```

위의 코드는 \listplatename 매크로로부터 'Plates'라는 타이틀을 가진 목차를 식자하는 \listofplates 명령을 정의한다. 엔트리 명칭은 plate이고 파일 확장자는 lop이다. 일반적으로 plate 페이지에는 면주를 인쇄하지 않는다. \cftpagenumbersoff 명령은 페이지 번호를 목차에 식자하지 않도록 한다.

이 페이지들이 별지삽입되는 경우, 그 위치는 짝수쪽 면과 홀수쪽 면의 사이에 오게 된다. afterpage 패키지 [Car95]는 현재 페이지가 끝난 직후에 무엇인가를 지시하는 데 사용된다. 아래 보인 코드는 이 패키지가 제공하는 \afterpage 매크로를 사용하여 다음 번 짝수 또는 홀수쪽 페이지 조판이 완료된 후에 어떤 일을 하여야 하는가를 지정하는 두 개의 매크로를 정의한다.

```
\newcommand{\afternextverso}[1]{%
  \afterpage{\ifodd\c@page #1\else\afterpage{#1}\fi}}
\newcommand{\afternextrecto}[1]{%
  \afterpage{\ifodd\c@page\afterpage{#1}\else #1\fi}}
```

\pageref{⟨*labelid*⟩} 명령은 문서의 특정 위치에 \label{⟨*labelid*⟩}로 지정된 페이지에 해당하는 번호를 식자해준다. 다음 코드는 \pageref로 주어지는 페이지 번호[3]를 앞이나 뒤에 찍어준다.

```
\newcounter{mempref}
\newcommand{\priorpageref}[1]{%
  \setcounter{mempref}{\pageref{#1}}\addtocounter{mempref}{-1}\thempref}
\newcommand{\nextpageref}[1]{%
  \setcounter{mempref}{\pageref{#1}}\addtocounter{mempref}{1}\thempref}
```

이러한 사전조치 후에 우리는 plate로 삽입될 부분을 제어하는 코드를 작성할 수 있다.

```
\afternextverso{\label{tip}
  \addtocontents{lop}{%
    Between pages \priorpageref{tip} and \pageref{tip}
    \par\vspace*{\baselineskip}}
  \addcontentsline{lop}{plate}{First plate}
  \addcontentsline{lop}{plate}{Second plate}
  ...
```

[3]아라비아 숫자인 경우만 가능함.

```
    \addcontentsline{lop}{plate}{Nth plate}
  }
```

plate 바로 다음 번의 홀수쪽이 오기 전까지 기다리기를 시작하면서, tip이라는 label을 삽입한다. \addtocontents 매크로는 그 인자를 lop 파일에 plate 리스트로 써넣는다. 또한 plate 페이지 전후의 페이지 번호도 함께 기록한다. plate가 나중에 책의 중간에 별지로 삽입되면 \caption은 사용할 수 없어도 \addcontentsline 매크로를 이용하여 plate title을 lop 파일에 써넣는 것은 가능하다.

위의 코드를 조금 수정하면 물리적으로 별지를 삽입하는 것이 아니라 전자적으로 삽입하도록 하는 것도 가능하다. 다만 면주와 \caption은 인쇄되지 않아야 한다.

10.5 그밖에

옛날 소설책이나, 최근의 교과서[4]에 보면 장 표지나 ToC, 혹은 양쪽 모두에 그 장의 내용에 관한 짧은 개요를 두고 있는 경우가 있다.

\chapterprecis{⟨*text*⟩}

\chapterprecis는 그 인자를 그 위치에 식자하면서 .toc에도 추가한다.

```
...
\chapter{}  first chapter
\chapterprecis{Our hero is introduced; family tree; early days.}
...
```

\chapterprecishere{⟨*text*⟩}
\chapterprecistoc{⟨*text*⟩}

\chapterprecis 명령은 이 두 명령을 호출하여 ⟨*text*⟩를 문서에 삽입(\chapterprecishere)하고, ToC에 넣는다(\chapterprecistoc). 필요하다면 각각 별도로 호출할 수 있다.

\prechapterprecis \postchapterprecis

\chapterprecishere 매크로는 \chapter가 불린 직후에 사용되어야 한다. ⟨*text*⟩ 인자는 quote 환경의 기울인 글꼴로 식자된다. 이 매크로의 정의는 다음과 같다.

[4] 예를 들면, Robert Sedgewick, *Algorithms*, Addison-Wesley, 1983.

```
\newcommand{\chapterprecishere}[1]{%
  \prechapterprecis #1\postchapterprecis}
```

여기서 \prechapterprecis와 \postchapterprecis는 다음과 같이 정의되어 있다.

```
\newcommand{\prechapterprecis}{%
  \vspace*{-2\baselineskip}%
  \begin{quote}\normalfont\itshape}
\newcommand{\postchapterprecis}{\end{quote}}
```

\prechapterprecis와 \postchapterprecis 매크로는 기본 모양이 아닌 다른 조판 스타일이 필요할 때 이를 변경하여 사용할 수 있다.

\precistoctext{⟨*text*⟩} \precistocfont

\chapterprecistoc 매크로는 \precistoctext 매크로를 toc 파일에 넣는다. 기본 정의는 다음과 같다.

```
\DeclareRobustCommand{\precistoctext}[1]{%
  {\leftskip \cftchapterindent\relax
   \advance\leftskip \cftchapternumwidth\relax
   \rightskip \@tocrmarg\relax
   \precistocfont #1\par}}
```

결과적으로, ToC에서 \precistoctext는 그 인자를 \precistocfont (기본값은 \itshape) 글꼴로 장 제목처럼 식자한다.

가끔은 특정 엔트리에 해당하는 전역 파라미터를 변경하고 싶을 때가 있다. 예컨대 그림을 책의 표지면(앞표지 또는 뒷표지의 안쪽 면)에 놓는 경우가 그러한데, 이 때는 LoF의 페이지 번호를 예컨대, '앞표지 안쪽'이라고 적어주어야 한다. 만약 '앞표지 안쪽'이 보통 페이지 번호처럼 식자되면 마진으로 먹어들어갈 것이다. 그러므로 이 특정 엔트리에 대해서는 파라미터 값을 바꾸어 두어야 한다.

\cftlocalchange{⟨*ext*⟩}{⟨*pnumwidth*⟩}{⟨*tocrmarg*⟩}

\cftlocalchange 명령은 엔트리를 ⟨*ext*⟩라는 확장명을 가진 파일에 쓰면서 \@pnumwidth, \@tocrmarg 전역 파라미터 길이값을 재설정한다. 이 명령은 특정 엔트리 이후에 달시 불리워져야 하는데 그래야 파라미터 값들을 원래 값으로 되돌릴 수 있기 때문이다. 인자에 풀리는 명령이 온다면 반드시 protect해주어야 한다.

\cftaddtitleline{⟨*ext*⟩}{⟨*kind*⟩}{⟨*text*⟩}{⟨*page*⟩}
\cftaddnumtitleline{⟨*ext*⟩}{⟨*kind*⟩}{⟨*num*⟩}{⟨*title*⟩}{⟨*page*⟩}

\cftaddtitleline 명령은 \contentsline 엔트리를 ⟨*ext*⟩에 쓰고 ⟨*kind*⟩ 엔트리에 ⟨*title*⟩ 표제와 ⟨*page*⟩ 페이지번호를 기록한다. 인자로 풀리는 명령이 오면 protect해주어야 한다. 즉, 하나의 엔트리는 다음 형태로 이루어져 있다.

```
\contentsline{kind}{title}{page}
```

\cftaddnumtitleline 명령은 \cftaddtitleline과 비슷하지만 ⟨*num*⟩을 \numberline의 인자로 취하도록 한다는 점이 다르다. 즉, 엔트리는 다음과 같은 형태로 이루어진다.

```
\contentsline{kind}{\numberline{num} title}{page}
```

이 명령을 사용하는 보기를 들어보자. \@pnumwidth와 \@tocrmarg에 해당하는 LaTeX 기본값이 각각 1.55em과 2.55em임을 상기하자. 표지 내지에 오는 그림을 다음과 같이 처리할 수 있다.

```
...
 this is the frontispiece page with no number
 draw or import the picture (with no \caption)
\cftlocalchange{lof}{4em}{5em} % make pnumwidth big enough for
                               % frontispiece and change margin
\cftaddtitleline{lof}{figure}{The title}{frontispiece}
\cftlocalchange{lof}{1.55em}{2.55em} % return to normal settings
\clearpage
...
```

.lof에 엔트리를 적어넣는 데 \caption이 필요하였지만, 여기서는 그렇지 않다. 캡션이 필요하다면 직접 만들거나, 보통 캡션 모양으로 상관없다면 나중에 설명하는 \legend 명령을 사용하라. 만약 삽화에 번호가 붙어 있다면, \cftaddtitleline 대신 \cftaddnumtitleline을 사용한다.

10.6 책의 마지막에 오는 요소들

참고문헌 목록과 색인을 만드는 명령은 표준 클래스의 것과 동일하다. 다만 두 가지가 다르다.

참고문헌 목록

```
\begin{thebibliography}{⟨exlabel⟩}
\bibitem ...
\end{thebibliography}
\bibname
```

bibliography는 `thebibliography` 환경으로 식자된다. 이 환경은 ⟨*exlabel*⟩이라는 인자를 하나 취하는데, 이것은 bibliography에서 가장 넓은 label의 폭을 나타내기 위해서 주는 문자이다. `\bibname` (기본값은 'Bibliography'이다) 값은 표제로 사용된다.

```
\bibintoc \nobibintoc
```

`\\bibintoc` 선언은 `thebibliography` 환경에 ToC 타이틀을 추가한다. `\nobibintoc`는 이 표제가 ToC에 추가되지 않도록 한다. 기본값은 `\bibintoc`이다.

```
\prebibhook \postbibhook
```

`\prebibhook`과 `\postbibhook`은 bibliography 표제가 식자된 후와 엔트리가 식자된 후에 각각 불린다. 기본값은 아무것도 하지 않는 것인데, 이들을 사용하여 bibliography 전체에 어떤 내용을 추가할 수 있다.

```
\renewcommand{\postbibhook}{%
CTAN is the Comprehensive TeX Archive Network and URLS for the
several CTAN mirrors can be found at \url{http://www.tug.org}.}
```

```
\setbiblabel{⟨style⟩}
```

문헌 목록 엔트리를 표시하는 label 스타일은 `\setbiblabel` 명령으로 설정한다. 기본 설정은 다음과 같다.

```
\setbiblabel{[#1]\hfill}
```

여기서 `#1`은 인용 기호 위치로서, 대괄호로 둘러싸인 숫자가 flushleft로 식자된다. 숫자 다음에 점만을 찍으려면 다음과 같이 한다.

```
\setbiblabel{#1.\hfill}
```

bibliography 환경은 다음과 같이 정의되어 있다.

```
\newenvironment{thebibliography}[1]{%
  \bibsection
  \begin{bibitemlist}{#1}}{\end{bibitemlist}\postbibhook}
```

\bibsection

\bibsection 매크로는 thebibliography 환경의 헤딩을 설정한다. 즉, 실제 문헌 목록이 열거되기 전에 이 매크로의 내용이 온다. 기본 정의는 다음과 같다.

```
\newcommand{\bibsection}{%
  \chapter*{\bibname}
  \bibmark
  \ifnobibintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\bibname}
  \fi
  \prebibhook}
```

bibliography의 헤딩 스타일을 바꾸려면 \bibsection을 재정의한다. 예를 들어, bibliography가 chapter 수준이 아니라 번호붙은 section 제목으로 식자되게 하고 싶으면 다음과 같이 한다.

```
\renewcommand{\bibsection}{%
  \section{\bibname}
  \prebibhook}
```

natbib이나 chapterbib 패키지 [Dal99, Ars01b]의 sectionbib 옵션을 사용하고 있다면 \bibsection 명령은 번호붙은 section으로 적절하게 바뀔 것이다.

jurabib [Ber02] 패키지는 독자적인 목록 열거 방법을 구현하기 위하여 thebibliography 환경을 재정의하고 있다. 그러나, 그 결과 Bibliography를 Table of Contents에 추가하는 것, 약간의 설명적 문장을 삽입하는 것 등을 할 수 없게 만든다. 이 기능을 복구하여 사용하고 싶다면 preamble에서 jurabib을 로드한 다음에 다음을 추가하라.

```
\usepackage{jurabib}
\makeatletter
\renewcommand{\bib@heading}{\bibsection}
\makeatother
```

그렇지만, Jens Berger 씨는 친절하게도, jurabib이 0.6 이후 버전에서 이러한 수정이 불필요하게 만들어 주었다.

\bibitemsep

natbib 패키지는 \bibsep이라는 길이 변수를 제공하는데, 이것은 bibliography의 엔트리 간의 간격을 변경하는 데 사용된다. \bibsep을 0pt로 하면 엔트리 간의 추가 간격이 없어진다. 같은 역할을 하는 길이변수를 이 클래스도 제공한다. \bibitemsep이 그것인데, 기본값은 \itemsep 값으로 지정되어 있다.

문헌목록은 list로 식자되므로, 아이템 간의 간격은 (\bibitemsep + \parsep)이다. 추가 간격을 없애려면

```
\setlength{\bibitemsep}{-\parsep}
```

위와 같이 한다.

\biblistextra

\biblistextra라는 명령은 bibliography 리스트가 설정된 직후에 불린다. 기본값은 아무것도 하지 않는 것이지만, 예를 들어 모든 bibliography 엔트리가 flushleft 정렬되도록 하기 위해, 아래 보인 것과 같이 list 파라미터를 수정하는 데 사용할 수 있다.

```
\renewcommand{\biblistextra}{%
  \setlength{\leftmargin}{0pt}%
  \setlength{\itemindent}{\labelwidth}%
  \setlength{\itemindent}{\labelsep}%
}
```

찾아보기

인덱스 관련 명령들은 표준 클래스와 비교해볼 때 현저히 개선되어 있으며 makeidx, showidx, index 패키지 [Bra94, Jon95]가 제공하는 기능들을 구현하고 있다. 따라서 이 패키지들을 함께 사용하면 안된다.

표준 클래스에서 인덱스는 2단으로 짜여진다.

\onecolindextrue \onecolindexfalse

\onecolindexfalse 선언(기본값)은 인덱스를 2단으로 조판한다. \onecolindextrue 선언은 그 이후의 인덱스를 1단으로 조판한다. 이것은 예를 들면 시의 첫 줄로 인덱스를 만들 때 유용하다.

\makeindex[⟨*file*⟩]
\printindex[⟨*file*⟩]

\makeindex 매크로는 preamble에 놓여야 한다. 이 매크로는 jobname.idx 라는 이름의 idx 파일을 연다. 여기서 jobname은 메인 LaTeX 소스 파일의 이름이다. 만약 ⟨*file*⟩ 선택 인자가 주어지면 file.idx 파일을 연다. \printindex 매크로는 theindex 환경과 그 속의 아이템들을 포함하고 있는 jobname.ind라는 ind 파일을 읽는다. ⟨*file*⟩ 선택 인자가 주어져 있으면 file.ind를 읽는다. MAKEINDEX 프로그램은 idx 파일을 ind 파일로 변환하는 역할을 한다.

\begin{theindex} entries \end{theindex}
\indexname

인덱스는 theindex 환경으로 2단 조판된다. 인덱스의 표제 명칭은 \indexname (기본값은 'Index') 로 주어진다.

\indexintoc \noindexintoc

\indexintoc 선언은 theindex 환경을 ToC에 추가한다. 반면 \noindexintoc는 표제를 ToC에 추가하지 않는다. 기본값은 \indexintoc이다.

\indexcolsep
\indexrule

\indexcolsep 길이값은 두 단 사이의 간격을 지정하는 것으로, 기본값은 35pt이다. 길이값 \indexrule은 단 사이에 그려넣는 수직선의 두께를 지정한다. 기본값은 0pt이다.

\preindexhook

\preindexhook은 표제가 식자된 후, 2단으로 항목 나열이 시작되기 전에 불린다. 기본값은 아무것도 하지 않는 것이지만 바꿀 수 있다. 다음 예를 보라.

```
\renewcommand{\preindexhook}{Bold page numbers are used to indicate
the main reference for an entry.}
```

\index[⟨*file*⟩]{⟨*item*⟩} \specialindex{⟨*file*⟩}{⟨*counter*⟩}{⟨*item*⟩}

\index 매크로는 ⟨*item*⟩ 인자를 idx 파일에 쓴다. 옵션 인자 ⟨*file*⟩이 주어져 있으면 file.idx에, 그렇지 않으면 jobname.idx에 쓴다. ⟨*item*⟩의 페이지도 idx 파일에 함께 기록된다. \specialindex 매크로는 ⟨*item*⟩ 인자와 해당 페이지(괄호 안에)를 file.idx에 쓰고, ⟨*counter*⟩도 함께 기록한다. 즉, 인덱스작성은 페이지 번호 이외의 것도 참조할 수 있다는 뜻이다. 그러나 hyperref 패키지를 사용하는 경우 special index가 ⟨*counter*⟩에 따라 표시됨에도 불구하고 페이지로 링크될 것이다. 예를 들면 그림 번호가 인덱스에 열거되면 hyperref 링크는 그 그림 자체가 아니라 그림이 나타난 페이지로 링크된다.

\see{⟨*item*⟩} \seename \seealso{⟨*items*⟩} \alsoname

\see 매크로는 \index 명령 안에서 사용될 수 있다. '⟨*item*⟩을 보라'라는 지시어를 페이지 숫자 대신 알려주는 것이다. 마찬가지로, \seealso 매크로는 독자에게 '⟨*items*⟩들도 함께 볼 것'이라고 지시한다.

```
\index{Alf|see{Alfred}}
\index{Frederick|seealso{Fred, Rick}}
```

'see'와 'see also'의 실제 값은 \seename과 \alsoname 매크로로 주어지는 것으로, 다음과 같이 정의되어 있다.

```
\newcommand{\seename}{see}
\newcommand{\alsoname}{see also}
```

\reportnoidxfilefalse \reportnoidxfiletrue

\reportnoidxfilefalse 선언이 이루어지면 LaTeX은 \makeindex로 선언되지 않은 idx 파일을 사용하지 않고 그냥 지나간다. 이것이 기본값이다. \reportnoidxfiletrue를 선언하면 LaTeX은 열리지 않은 파일에 쓰기를 시도하느라고 불평을 할 것이다.

\showindexmarktrue \showindexmarkfalse

\showindexmarktrue를 선언하면 모든 \index와 \specialindex, ⟨*item*⟩ 인자들이 index 명령이 주어진 페이지의 여백에 열거된다. 기본값은 \showindexmarkfalse이다.

찾아보기 만들기와 natbib 패키지

natbib [Dal99] 패키지를 올린 후에 `\citeindextrue`를 preamble에 써주면 인용 인덱스를 만들어준다.

`\citeindexfile`

인용 인덱스에 이용되는 파일의 이름은 `\citeindexfile`에 저장되어 있는데 기본적으로 다음과 같이 정의되어 있다.

```
\newcommand{\citeindexfile}{\jobname}
```

즉, 인용 엔트리들은 기본 idx 파일에 저장된다. 원한다면 다음과 같이 바꿀 수 있다.

```
\renewcommand{\citeindexfile}{names}
```

`\citeindexfile`을 바꾼다면 다음을 두어야 한다.

```
\makeindex[\citeindex]
```

이것은 아래 설정 0이후에 온다.

```
\usepackage[...]{natbib}
```

결과적으로, 다음과 같은 두 가지 선택이 가능하다. 하나는,

```
\renewcommand{\citeindexfile}{authors} % write to authors.idx
\makeindex[\citeindexfile]
\usepackage{natbib}
\citeindextrue
...
\renewcommand{\indexname}{Index of citations}
\printindex[\citeindexfile]
```

위와 같이 하는 것이고, 다른 하나는 다음처럼 하는 것이다.

```
\usepackage{natbib}
\citeindextrue
\makeindex
...
\printindex
```

idx 파일 범용화

표준 클래스에서 인덱스 아이템은 idx 파일에 직접 쓴다. 그러나 이 클래스에서는 aux 파일에 먼저 내용을 기록하였다가, 그 다음번 LaTeX이 실행될 때 aux로부터 idx 파일이 만들어지도록 설계되어 있다.

이러한 이 단계 방식이 가지는 결점은 인덱스 아이템을 변경한 후에는 LaTeX을 적어도 두 번 실행해주어야 원하는 결과가 idx에 반영된다는 것이다. 그러나 사실은 BibTeX으로 bibliography를 만들 때도 이와 비슷한 과정을 거친다.

이 방식의 장점은 \include로 포함된 파일에서 얻어진 인덱스 아이템이 유실되지 않는다는 것이다. 표준 클래스의 idx에 직접 쓰는 방식은 'include되지 않은' 인덱스 아이템이 나타나지 않는다.

11 캡션과 플로트

이 장은 *hangnum* 장 스타일을 사용한다. 절 번호는 `\hangsecnum` 선언에 의하여 여백까지 빠져나가도록 한다.

11.1 들어가는 말

일부 편집자나 저자들은 표준 LaTeX과는 다른 형태의 캡션 스타일을 요구하는 경우가 있다. 이 장에서는 사용자의 요구에 맞는 캡션 스타일을 어떻게 구현하는지를 설명하겠다.

여러 개의 표에 같은 캡션 번호를 붙이고 첫번째 표를 제외한 모든 이어지는 표에는 이어지는 캡션 여러 부분 표(multi-part table)을 사용해야 하는 경우도 있다. 이 경우 그 표는 저자에 의해 연속 테이블임이 명시되어야 하는데, 이 클래스에서는 간단히 'continuation' 캡션 명령을 사용하여 이러한 요구를 충족하도록 한다. 또, '무기명' 캡션 명령을 어떤 플로트 환경에서도 사용할 수 있도록 기능을 확장하였다. 캡션은 플로트가 아닌 경우에도 사용할 수 있도록 정의할 수 있다. 예를 들면 minipage 안에 그림을 둔 다음, 이것이 보통 figure 환경 안에 놓인 것처럼 캡션을 달 수도 있다. 그리고, 새로운 플로트 환경을 정의할 수도 있게 하고 있다.

아래에서 설명하는 명령과 환경은 ccaption 패키지 [Wil01d]에서 제공하는 것과 매우 비슷하다.

11.2 캡션

캡션 스타일 바꾸기

`\captiondelim{`⟨*delim*⟩`}`

기본 캡션 스타일은 캡션 번호와 캡션 타이틀 사이에 콜론을 두는 것이다. `\captiondelim` 명령은 이 경계 문자를 바꾸려 할 때 사용한다. 예를 들면, en-대시를 콜론 대신 두려 한다면, `\captiondelim{-- }`라고 하면 된다. 만약 ⟨*delim*⟩ 인자가 주어지지 않으면 경계문

자와 타이틀 사이에 공백을 주지 않고 붙어서 출력된다는 사실을 알아두자. 이 클래스의 초기 기본값은 일반적인 캡션의 경우와 같이 \captiondelim{: }로 하는 것이다.

\captionnamefont{⟨*fontspec*⟩}

\captionnamefont 명령은 ⟨*fontspec*⟩을 설정한다. 이것은 캡션 명칭을 식자하는 데 사용된다. 즉, 캡션의 첫부분에서 경계문자까지(예를 들면 '표 3:' 부분이다)를 식자할 글꼴을 지정한다. ⟨*fontspec*⟩에는 폰트 지정 명령과 텍스트가 올 수 있다. 캡션의 첫부분은 보통 다음과 같이 구성된다. {\captionnamefont Table 3: }. 그러므로, 글꼴 설정을 바꾸려면 text-스타일의 명령 대신, 글꼴 선언형을 써야 한다. \captionnamefont{\Large\sffamily} 처럼 지정하면 large san-serif 폰트를 사용하도록 하는 것이다. 이 클래스의 초기값은 normal 글꼴을 쓰도록 \captionnamefont{} 로 설정되어 있다.

\captiontitlefont{⟨*fontspec*⟩}

마찬가지로, \captiontitlefont는 ⟨*fontspec*⟩을 설정하는데, 이것은 캡션의 타이틀 텍스트를 식자하는 데 사용된다. 예를 들면, \captiontitlefont{\itshape}로 하면 타이틀 텍스트가 이탤릭체로 식자된다. 이 클래스의 초기값은 \captiontitlefont{}로 하여 normal 글꼴을 쓰는 것이다.

\captionstyle[⟨*short*⟩]{⟨*normal*⟩}
\raggedleft \centering \raggedright centerlastline

캡션 스타일은 \captionstyle 선언에 의해 설정된다. ⟨*short*⟩ 옵션 인자가 없으면 모든 캡션은 ⟨*normal*⟩에 따른다. 옵션 인자 ⟨*short*⟩가 있으면, 한 줄 이하 길이를 가진 캡션은 ⟨*short*⟩에 의해 식자된다. 기본값은 캡션이름과 타이틀 텍스트가 들여밀기 없는 상자형 문단으로 조판되는 것이다.

\raggedleft, \centering, \raggedright, \centerlastline 들이 인자로 올 수 있는 유효한 값이다. 이 클래스는 초기값으로,

```
\captionstyle{}
```

와 같이 설정해두고 있는데, 이것은 상자형 문단 스타일을 지정하는 것이다. \centerlastline 스타일은 블럭 문단 형태이기는 하나 마지막 줄이 중앙정렬된다.

\hangcaption
\indentcaption{⟨*length*⟩}
\normalcaption

\hangcaption 명령은 여러 줄 캡션 타이틀의 두번째 이후 줄이 캡션 이름 만큼 들여밀기 되는 내어밀기 문단이 되도록 한다. \indentcaption 명령은 타이틀 행의 첫 줄을 ⟨*length*⟩ 만큼 들여밀기한다. 이 명령들은 \captionstyle{...}과는 독립적이다. 캡션 들여밀기와 내어밀기를 동시에 할 수는 없다. \normalcaption 명령은 이전의 \hangcaption이나 \indentcaption 명령을 취소한다. 이 클래스의 초기값은 \normalcaption을 선언하여 들여밀기 없는 문단 형식으로 만드는 것이다.

```
\changecaptionwidth
\captionwidth{⟨length⟩}
\normalcaptionwidth
```

\changecaptionwidth 명령을 주면 \captionwidth에 의해 주어진 ⟨*length*⟩ 만큼의 길이가 캡션 전체 길이가 되도록 한다. \normalcaptionwidth는 일반적인 full width 캡션을 만든다. 이 클래스에서는 \normalcaptionwidth로 하고 \captionwidth{\linewidth}로 그 길이를 초기 지정하고 있다. 캡션이 sidecap 패키지 [NG98]에 의해 사이드 캡션 안에 놓이게 하려면 반드시 \normalcaptionwidth 캡션이어야 한다.

```
\precaption{⟨pretext⟩}
\postcaption{⟨posttext⟩}
```

\precaption 명령과 \postcaption 명령은 ⟨*pretext*⟩와 ⟨*posttext*⟩를 설정한다. 각각 캡션 만들기 전과 후에 처리되는 것이다. 예를 들면

```
\precaption{\rule{\linewidth}{0.4pt}\par}
\postcaption{\rule{\linewidth}{0.4pt}}
```

와 같이 하면 캡션의 아래와 위에 선이 그어질 것이다. 이 클래스의 초기값은 \precaption{}, \postcaption{}으로 설정하여 보통 모양을 가지도록 하고 있다.

위의 명령이 플로트 안이나 특정 환경 안에서 쓰이면 그 효과는 환경 범위 안으로 한정된다. Preamble이나 본문 텍스트에서 사용하면 그 효과는 파라미터 값을 바꾸기 전까지 계속된다. 이 명령들은 'List of...'내에 들어갈 타이틀의 형태를 바꾸지 않는다.

```
\\[⟨length⟩]
\\*[⟨length⟩]
```

일반적인 LaTeX 명령 \\는 캡션 안에서 새로운 줄을 시작하게 하는 데 쓰인다. 그러나 \\이 풀리는 명령임을 명심하라. 따라서 'List of...'에 텍스트를 추가하려면 이 명령을 protect해야 한다.

표 11.1
표 CAPTION 스타일의 재설계

three	III
five	V
eight	VIII

```
\caption{Title with a \protect\\ new line in both the body and List of}
\caption[List of entry with no new line]{Title with a \\ new line}
\caption[List of entry with a \protect\\ new line]{Title text}
```

결과적으로 캡션은 다음과 같은 형태로 식자된다.

```
\precaption
{\captionnamefont NAME NUMBER \captiondelim}
{\captionstyle\captiontitlefont THE TITLE}
\postcaption
```

위의 명령의 기본값을 사용하면 단순히 다음과 같은 형태로 만들어진다.
`{NAME NUMBER: }{THE TITLE}`

스타일 설정 명령으로는, 캡션의 모양을 가볍게 바꾸는 것 이외에도 상당한 변경을 가하는 것이 가능하다. 이름과 타이틀이 sans 폰트로 식자되도록 캡션 스타일을 바꾸려면 다음과 같이 하는 것으로 충분하다.

```
\captionnamefont{\sffamily}
\captiontitlefont{\sffamily}
```

좀더 현저한 스타일의 변화를 표 [11.1]에 보였다. 코드는 다음과 같다.

```
\begin{table}
\centering
\captionnamefont{\sffamily}
\captiondelim{}
\captionstyle{\\}
\captiontitlefont{\scshape}
\setlength{\belowcaptionskip}{10pt}
\caption{Redesigned table caption style} \label{tab:style}
\begin{tabular}{lr} \hline
 ...
\end{table}
```

위의 설정으로 캡션 형식은 대강 다음처럼 된다 (\centering 범위 안에서 처리되고 있다).

```
{\sffamily NAME NUMBER}{\\ \scshape THE TITLE}
```

개행 명령(\\)이 이 형식의 처음에 오면 안된다는 사실을 알아두자. 두번째 부분을 보면 \captionstyle{\\}을 이용해서 설정하고 \captiondelim{\\}를 이용하고 있지 않은 것을 볼 수 있다.

여러 가지 캡션 스타일들을 섞어쓰려 하는 경우 이러한 비표준적인 스타일에 대하여 특별한 캡션 명령을 정의하고 싶을 때가 있을 것이다. 예를 들면 표 [11.1]에 나타난 캡션 스타일에 대해서 다음과 같이 정의하는 방법이 있다.

```
\newcommand{\mycaption}[2][\@empty]{
  \captionnamefont{\sffamily\hfill}
  \captiondelim{\hfill}
  \captionstyle{\centerlastline\\}
  \captiontitlefont{\scshape}
  \setlength{\belowcaptionskip}{10pt}
  \ifx #1\@empty \caption{#2}\else \caption[#1]{#2}}
```

NOTE: @ 기호를 포함하는 코드는 .sty 패키지 파일 안에 있거나 \makeatletter...\makeatother 짝으로 둘러싸야 한다.

표 [11.1]의 예제 코드는 이제 다음과 같이 쓸 수 있다.

```
\begin{table}
\centering
\mycaption{Redesigned table caption style} \label{tab:style}
\begin{tabular}{lr} \hline
 ...
\end{table}
```

\mycaption에 해당하는 코드에 두 개의 \hfill 명령과 \centerlastline을 추가한 점을 원래의 정의와 비교해보라. 원래의 정의는 한 줄짜리 캡션에 적당하지만 여러 줄 캡션에는 부적당하였음을 알 수 있을 것이다. 위와 같이 추가함으로써 한 줄의 경우와 여러 줄의 경우에 모두 적당하도록 고쳐짐으로써, 캡션 이름이 여러 줄 타이틀의 마지막 줄에서와 같이 중앙정렬되게 되었다. 이로써 좀더 균형잡힌 모습을 갖게 된다.

연속되는 캡션과 레전드

\contcaption{⟨*text*⟩}

표 11.2: A multi-part table

just a single line	1

표 11.2: Continued

just a single line	2

표 11.2: Concluded

just a single line	3

`\contcaption` 명령은 'continuation'이나 'concluded' 캡션을 플로트에 놓는 데 쓰인다. 플로트 번호를 증가시키지도 않고 플로트 목차에 엔트리를 추가하지도 않는다. 단지 앞서 사용된 `\caption` 명령의 번호를 반복한다.

표 [11.2]는 `\contcaption` 명령의 사용방법을 보여준다. 이 표는 다음 코드로 만들어진 것이다.

```
\begin{table}
\centering
\caption{A multi-part table} \label{tab:m}
\begin{tabular}{lc} \hline
 just a single line & 1 \\ \hline
\end{tabular}
\end{table}

\begin{table}
\centering
\contcaption{Continued}
\begin{tabular}{lc} \hline
 just a single line & 2 \\ \hline
\end{tabular}
\end{table}

\begin{table}
\centering
\contcaption{Concluded}
\begin{tabular}{lc} \hline
 just a single line & 3 \\ \hline
\end{tabular}
\end{table}
```

`\legend{`⟨*text*⟩`}`

표 11.3: Another table

A legendary table	5
with two lines	6

The legend

\legend 명령은 무기명 캡션을 플로트 환경에 붙이는 데 사용된다. 그러나 플로트가 아니라도 사용할 수 있다.

예를 들면, 다음 코드는 두 줄 짜리 표 [11.3]을 만든다. \legend 명령은 플로트 내에서 \caption 명령과는 무관하게 붙일 수 있다.

```
\begin{table}
\centering
\caption{Another table} \label{tab:legend}
\begin{tabular}{lc} \hline
 A legendary table & 5 \\
 with two lines    & 6 \\ \hline
\end{tabular}
\legend{The legend}
\end{table}
```

캡션 붙은 플로트는 일반적으로 table과 figure 환경이라고 생각할 수 있지만 다른 종류의 플로트도 있다. 좀더 재미있는 보기로, 다음 코드를 보자. 이것은 마진 노트에 제목을 붙여준다. 오른쪽 마진에 나와 있는 것이 그 결과이다.

레전드 제목

이것은 마진 노트에 레전드를 붙인 예입니다.

```
\marginpar{\legend{레전드 제목} 이것은 마진 노트에 레전드를 붙인
예입니다.}
```

만약 레전드 텍스트가 'List of...'에 포함되기를 원한다면 \addcontentsline 명령을 \legend 명령과 함께 사용하라. 다음에 보기를 보였다.

```
\addcontentsline{lot}{table}{Titling text} % left justifified
\addcontentsline{lot}{table}{\protect\numberline{}Titling text} % indented
```

첫번째 예는 레전드 텍스트의 첫번째 줄을 일반적인 table 번호 아래에 놓는다. 두번째 보기는 레전드 텍스트 첫 행을 일반적인 table 타이틀 아래에 둔다. 어떤 경우든 여러 줄 레전드 텍스트의 두번째 이후 행은 일반적인 타이틀 행 아래 놓일 것이다.

한 보기로서, Legendary table을 다음 코드로 만들어보자.

```
\begin{table}
```

Legendary table

An anonymous table	5
with two lines	6

```
\centering
\captiontitlefont{\sffamily}
\legend{Legendary table}
\addcontentsline{lot}{table}{Legendary table (toc 1)}
\addcontentsline{lot}{table}{\protect\numberline{}Legendary table (toc 2)}
\begin{tabular}{lc} \hline
 An anonymous table & 5 \\
 with two lines     & 6 \\ \hline
\end{tabular}
\end{table}
```

표 목차에 가서 \addcontentsline의 두 가지 형태가 어떤 모양인지 확인해보도록 하라.

\caption 명령의 경우와 마찬가지로, 레전드의 전후 공백은 \abovecaptionskip과 \belowcaptionskip 길이로 제어된다.

\namedlegend[⟨*short-title*⟩]{⟨*long-title*⟩}

일반적으로 \namedlegend 명령은 \caption 명령과 같다. 그러나 캡션에 번호를 붙이지 않으며 'List of...' 파일에 엔트리를 기록하지 않는 것이 기본값이라는 점이 다르다. \caption 명령과 마찬가지로 플로트 환경의 이름(즉, table 환경에 해당하는 \tablename)을 타이틀 텍스트 앞에 붙인다. 다음 보기는 *Named legendary table*의 소스 코드이다.

```
\begin{table}
\centering
\captionnamefont{\sffamily}
\captiontitlefont{\itshape}
\namedlegend{Named legendary table}
\begin{tabular}{lr} \hline
seven & VII \\
eight & VIII \\ \hline
\end{tabular}
\end{table}
```

\flegtype{⟨*name*⟩}
\flegtoctype{⟨*title*⟩}

표: *Named legendary table*

seven	VII
eight	VIII

`\flegtype` 매크로에서 `type`은 플로트 환경의 이름(예를 들면 `table`)인데, `\namedlegend` 매크로에 의하여 호출된다. ⟨*name*⟩이 `\namedlegend`에서 이름으로 사용되도록 정의해주는 역할을 한다. 두 가지의 기본값이 있다.

```
\newcommand{\flegtable}{\tablename}
\newcommand{\flegfigure}{\figurename}
```

이 값들은 `\renewcommand`에 의하여 바꾼다. `\flegtoctype` 매크로에서 `type`은 플로트의 이름이고, 마찬가지로 `\namedlegend` 매크로에 의하여 호출된다. ⟨*title*⟩을 'List of...'에 추가하도록 해준다. 기본값은 아무것도 정의하지 않은 것이며, `\renewcommand`로 바꿀 수 있다. 아래는 table에 해당하도록 바꾸는 예이다.

```
\renewcommand{\flegtoctable}[1]{
  \addcontentsline{lot}{table}{#1}}
```

`\legend` 명령은 번호붙지 않은 간단한 헤딩을 만든다. 그런데 가끔 플로트 환경에 해당하는 것을 minipage 안에 넣고 그 밖에 이름과 번호붙은 캡션을 붙여야 할 때가 있다. 테이블이나 그림을 문서의 특정한 위치에 반드시 두려고 할 때 그러하다.

`\newfixedcaption[`⟨*capcommand*⟩`]{`⟨*command*⟩`}{`⟨*float*⟩`}`
`\renewfixedcaption[`⟨*capcommand*⟩`]{`⟨*command*⟩`}{`⟨*float*⟩`}`
`\providefixedcaption[`⟨*capcommand*⟩`]{`⟨*command*⟩`}{`⟨*float*⟩`}`

`\newfixedcaption` 명령 및 그 아류들은 ⟨*command*⟩라는 새로운 캡션 명령을 만든다. 이것은 ⟨*float*⟩ 플로트 환경 밖에서 사용할 수 있다. ⟨*float*⟩ 환경과 캡션 명령 ⟨*capcommand*⟩은 모두 `\newfixedcaption`가 사용되기 전에 정의되어 있어야 한다. `\namedlegend`가 ⟨*capcommand*⟩로 사용될 수도 있다. `\renewfixedcaption`과 `\providefixedcaption` 명령은 `\newfixedcaption`과 동일한 인자를 취한다. 이 세 명령은 `\newcommand` 류의 명령과 사용법이 유사하다.

예를 들면 새로운 `\figcaption` 명령은 `figure` 환경 밖에서도 그림에 캡션을 붙일 수 있게 한다.

`\newfixedcaption{\figcaption}{figure}`

⟨*capcommand*⟩ 선택인자는 플로트 캡션 명령의 이름을 alias해준다. 기본값은 `\caption`

이다. 선택 인자가 사용되어야 하는 보기를 하나 들면, 새로운 연속 캡션 명령을 플로트가 아닌 테이블에 \ctabcaption이라는 이름으로 붙이고자 할 때 다음과 같이 한다.

```
\newfixedcaption[\contcaption]{\ctabcaption}{table}
```

\newfixedcaption에 의해 만들어지는 캡션 명령은 원래 ⟨*capcommand*⟩에서와 마찬가지로 이름과 번호가 붙게 된다. 그리고 여기에 \label을 붙일 수 있고 'List of...'의 적절한 위치에 표시된다. 이것은 플로트 환경 안에서도 쓸 수 있지만 'List of...'에 캡션에 붙이거나 'List of...'를 엔트리에 넣거나 할 때의 환경 이름을 사용할 수 없다. 예를 들면, \ctabcaption을 figure 안에서 사용하게 되면 여전히 **그림...** 이라는 이름이 붙는 캡션이 만들어진다.

가끔 그림이 있는 맞은편 페이지에 캡션을 붙여야 할 때가 있다. \newfixedcaption은 이럴 경우 유용하다. figure 캡션이 실제 그림이 있는 그 페이지나 그 바로 직전의 빈 페이지에 놓여야 하는 경우, 다음과 같이 할 수 있다.

```
\newfixedcaption{\figcaption}{figure}
 ...
\afterpage{ % fill current page then flush pending floats
  \clearpage
  \begin{midpage}  %  vertically center the caption
  \figcaption{The caption} %  the caption
  \end{midpage}
  \clearpage
  \begin{figure}THE FIGURE, NO CAPTION HERE\end{figure}
  \clearpage
}  % end of \afterpage
```

afterpage 패키지 [Car95]가 필요하다. 이 패키지는 LaTeX 기본 tools 묶음에 포함되어 있다. midpage 패키지는 midpage 환경을 제공하는데, 이것은 다음과 같이 간단히 정의되어 있다.

```
\newenvironment{midpage}{\vspace*{\fill}}{\vspace*{\fill}}
```

이 코드는 필요에 따라 조정해서 써야 한다. \cleartoevenpage 명령은 다음번 짝수쪽까지(\cleardoublepage가 다음번 홀수쪽에서 다시 시작하게 하고, \clearpage는 무조건 다음쪽에서 다시 시작하게 하는 것과 비교해볼 것) 새로운 페이지를 만든다.

다중언어 캡션

어떤 문서에서는 2개 국어(또는 그 이상) 캡션을 요구할 때가 있다. 이 클래스는 이 기능을 위한 일련의 명령을 제공한다. 이 명령을 확장하여 3개 국어 캡션을 붙이는 문제는 독자에게 연습문제로 남겨둔다.

EXAMPLE FIGURE WITH BITWONUMCAPTION

그림 11.1: 긴 \bitwonumcaption

Figure 11.1: Long \bitwonumcaption

EXAMPLE FIGURE WITH BIONENUMCAPTION

그림 11.2: 긴 한국어 \bionenumcaption

Figure 11.2: Long English \bionenumcaption

\bitwonumcaption[⟨*label*⟩]{⟨*short1*⟩}{⟨*long1*⟩}{⟨*NAME*⟩}{⟨*short2*⟩}{⟨*long2*⟩}
\bionenumcaption[⟨*label*⟩]{⟨*short1*⟩}{⟨*long1*⟩}{⟨*NAME*⟩}{⟨*short2*⟩}{⟨*long2*⟩}

2개 언어 캡션은 bitwonumcaption 명령으로 식자한다. 이 명령은 여섯 개의 인자를 가진다. 첫번째 선택인자 ⟨*label*⟩은 레이블의 이름이다. 필요하다면 이름을 지정한다. ⟨*short1*⟩과 ⟨*long1*⟩는 \caption 명령의 옵션 인자로 쓰는 것에 해당하는 짧은 캡션 텍스트와 긴 캡션 텍스트이다. ⟨*NAME*⟩ 인자의 값은 두번째 언어의 캡션 이름으로 쓰인다. ⟨*short2*⟩와 ⟨*long2*⟩는 두번째 언어의 짧고 긴 캡션 텍스트이다. 예를 들어, 첫번째 언어가 한국어이고 두번째 언어가 영어라면 그림은 다음과 같이 캡션이 주어져야 한다.

\bitwocaption{짧은 타이틀}{긴 타이틀}{Figure}{Short}{Long}

짧은 텍스트가 필요하지 않다면 인자를 비우거나 한두 개의 공백을 넣어둔다.

\bitwocaption[fig:bi1]{}{제목}{Figure}{ }{Long}

두 언어 텍스트는 모두 'List of...'에 들어간다. 그리고 모두 번호가 붙는다.

그림 11.1은 위의 코드를 이용한 예제이다.

\bionenumcaption 명령은 \bitwonumcaption과 동일한 인자를 취한다. 두 명령의 다른 점은 \bionenumcaption은 'List of...'에서 두번째 언어의 텍스트에 번호를 붙이지 않는다는 것이다. 그림 11.2는 \bionenumcaption을 이용한 예이다.

【memhangul】 원문에는 영어와 독일어의 예를 들고 있으나 번역본에서는 한국어와 영문의 예로 바꾸었다. ■

\bicaption[⟨*label*⟩]{⟨*short1*⟩}{⟨*long1*⟩}{⟨*NAME*⟩}{⟨*long2*⟩}

2개 언어 캡션을 \bicaption 명령으로 식자하면 두번째 언어 텍스트는 'List of...'에 들어가지 않는다. 이 명령은 다섯 개의 인자를 취한다. 선택 인자 ⟨*label*⟩은 필요하다면 레이블을 지정하는 것이다. ⟨*short1*⟩과 ⟨*long1*⟩은 문서 중심언어의 짧은 캡션과 긴 캡션 텍스트에 해당한다. ⟨*NAME*⟩ 인자의 값은 두번째 언어 캡션의 캡션 명칭이다. 마지막 인자인 ⟨*long2*⟩는 두번째 언어의 캡션 텍스트('List of...'에 들어가지 않는다)이다. 예를 들어 주요 언어가 한국어이고 두번째 언어가 영어일 때,

EXAMPLE FIGURE WITH A RULED BICAPTION

그림 11.3: 긴 캡션

Figure 11.3: English Caption

`\bicaption{짧은 제목}{긴 제목}{Figure}{LongCaption}`
과 같이 지정한다. 짧은 타이틀 텍스트가 필요없다면 이 인자를 비우거나 한두 개의 스페이스를 넣어둔다.

그림 11.3은 `\bicaption`을 이용한 예이다. 코드는 다음과 같다.

```
\begin{figure}
\centering
EXAMPLE FIGURE WITH A RULED BICAPTION
\precaption{\rule{\linewidth}{0.4pt}\par}
\midbicaption{\precaption{}\postcaption{\rule{\linewidth}{0.4pt}}}
\bicaption[fig:bi2]{짧은 한국어 \cs{bicaption}}{긴
캡션}{Figure}{English Caption}
\end{figure}
```

`\bicontcaption{`⟨*long1*⟩`}{`⟨*NAME*⟩`}{`⟨*long2*⟩`}`

2개 언어의 이어지는 캡션은 `\bicontcaption` 명령으로 식자한다. 이 경우에는 어느 것도 'List of...'에 들어가지 않는다. 이 명령은 세 개의 인자를 취하는데, ⟨*long1*⟩은 문서 주요 언어의 캡션 텍스트이다. ⟨*NAME*⟩ 인자의 값은 두번째 언어 캡션의 캡션 이름으로 사용된다. 마지막 인자인 ⟨*long2*⟩는 두번째 언어의 캡션 텍스트이다. 예를 들어 주요 언어가 한국어이고 두번째 언어가 독일어라면, 다음과 같이 한다.
`\bicontcaption{표 계속}{Bild}{Fortgefahren}`

`\midbicaption{`⟨*text*⟩`}`

2개 언어 캡션은 `\caption` 명령을 각각의 언어로 두 번 불러서 구현하였다. `\midbicaption` 명령은 `\precaption`과 `\postcaption` 명령이 하는 일과 비슷한데, 두번째 `\caption`을 부르기 전에 실행된다. 이 명령으로는 첫번째 캡션과 대조되는 두번째 캡션의 모양을 설정할 때 쓰일 수 있다. 예컨대 일반적인 캡션의 아래위에 줄을 친 경우라면 두 캡션 사이에 두 줄이 그어지는 것은 바람직한 상황이 아닐 것이다. 그러므로 2개 언어 캡션에서 다음과 같이 하여 이런 상황을 피할 수 있다.

```
\precaption{\rule{\linewidth}{0.4pt}\par}
\postcaption{}
\midbicaption{\precaption{}\postcaption{\rule{\linewidth}{0.4pt}}}
```

여기서는 두 개의 캡션 중 첫번째 것 앞에 선을 그었다. 그런 다음에 `\midbicaption`... 명령으로 pre-caption 선을 없애고 post-caption 선을 두번째 캡션에 추가한다. 이 클래스는 `\midbicaption{}`으로 초기화해두고 있다.

서브캡션

subfigure 패키지 [Coc02]는 큰 figure나 table 안에 sub-figure 또는 sub-table을 두고 캡션을 붙일 수 있도록 한다. subfigure 패키지는 이 클래스와 함께 사용할 수도 있지만 아래 설명하는 이 클래스 자신이 제공하는 명령을 쓸 수도 있다. 이 명령들은 subfloat[1]가 정의된 플로트 환경 안에서만 사용할 수 있다.

`\subcaption[`⟨*list-entry*⟩`]{`⟨*subcaption*⟩`}`

`\subcaption` 명령은 `\caption` 명령과 비슷하다. ⟨*subcaption*⟩으로 주어지는 지정자를 식자한다. 이 지정자는 괄호로 묶어진 알파벳 글자로 이루어진다. ⟨*list-entry*⟩ 선택 인자가 주어지면, ⟨*list-entry*⟩가 목차에 추가된다. `\subcaption` 매크로는 `minipage` 환경과 같은 고정폭 박스로 취급되는데 이 때 이 폭은 `minipage` 환경 자체의 폭과 같다.

예를 들어보자.

```
\begin{figure}
  \centering
  \begin{minipage}{0.3\textwidth}
    \verb?Some verbatim text?
    \subcaption{First text}
  \end{minipage}
  \hfill
  \begin{minipage}{0.3\textwidth}
    \verb?More verbatim text?
    \subcaption{Second text}
  \end{minipage}
  \caption{Verbatim texts}
\end{figure}
```

[1] 182 쪽의 §11.3을 보라.

만약 figure가 sub-figure를 포함하고 있으면서 이어지고 있다면 sub-figure의 캡션도 이어지도록 하는 것이 좋을 것이다. 예컨대, 그림 3이 세 개의 sub-figure, A, B, C를 포함하고 있는데, 같은 그림 3으로 계속 이어가고 싶다면 sub-figure들의 캡션은 D, E 등이 되어야 한다.

```
\contsubcaption[⟨list-entry⟩]{⟨subcaption⟩}
```

`\contsubcaption` 명령은 `\subcaption` 명령의 'continued' 버전이다.

```
\label(⟨bookmark⟩){⟨key⟩}
\subcaptionref{⟨key⟩}
```

`\label` 명령은 `\subcaption`이나 `\contsubcaption` 명령의 ⟨*subcaption*⟩에 포함될 수 있다. `\ref`를 사용하여 이 레이블을 참조하면 플로트의 번호(`\caption` 명령으로 레이블이 주어지면 얻을 수 있다)와, subcaption 지정자가 식자된다. 반면 `\subcaptionref` 명령을 쓰면 subcaption 지정자만이 식자된다.

hyperref 패키지를 사용하는 경우, ⟨*subcaption*⟩ 인자 안에서 사용되는 `\label` 명령은 각진괄호가 아니라 괄호로 둘러싸여지는, ⟨*bookmark*⟩라는 선택 인자를 하나 취한다. 이 인자는 'Subfigure 3.2(c)' 형식의 북마크 필드를 생성한다.

```
\tightsubcaptions \loosesubcations
```

`\tightsubcaptions` 선언은 subcaption 전후에 공백을 거의 두지 않는다. 반면 `\loosesubcaptions`는 공백을 좀더 넉넉하게 둔다. 기본값은 `\tightsubcaptions`이다.

```
\subcaptionsize{⟨font-size⟩}
\subcaptionlabelfont{⟨fontspec⟩}
\subcaptionfont{⟨fontspec⟩}
```

subcaption에 사용되는 폰트 크기는 `\subcaptionsize`로 지정한다. 기본값은 다음과 같다.

```
\subcaptionsize{\footnotesize}
```

식별자와 subcaption에 사용되는 폰트는 `\subcaptionlabelfont`와 `\subcaptionfont`로 지정한다. ⟨*fontspec*⟩은 하나 이상의 폰트 스타일 선언(예, `\bfseries\slshape`)이 올 수 있다. 디폴트는 다음과 같다.

```
\subcaptionlabelfont{\normalfont}
\subcaptionfont{\normalfont}
```

\subcaptionstyle{⟨*style*⟩}

기본적으로 식별자와 subcaption 타이틀은 들여밀기 없는 상자형 문단으로 식자된다. \subcaptionstyle은 이것을 바꿀 수 있게 해준다. ⟨*style*⟩ 인자로 올 수 있는 값은 다음과 같다. \centering, \raggedleft, \raggedright. \centerlastline 스타일은 상자형 문단으로 식자하지만 마지막 행을 중앙정렬한다. 클래스 초기값은 \subcaptionstyle{}로 되어 있어서 보통의 상자형 문단 스타일을 지정하고 있다.

\hangsubcaption \shortsubcaption \normalsubcaption

\hangsubcaption 선언은 subcaption이 식별자를 타이틀 위에 두도록 식자하게 한다. \shortsubcaption 선언은 길이가 한 줄을 넘지 않을 경우 중앙정렬이 아닌 왼쪽 정렬이 되도록 한다. \normalsubcaption 명령은 \hangsubcaption이나 \shortsubcaption에 의한 설정을 취소하고 기본 모양으로 되돌린다. 클래스의 초기값도 \normalcaption이다.

\subtop[⟨*list-entry*⟩][⟨*subcaption*⟩]{⟨*text*⟩} \subbottom[⟨*list-entry*⟩][⟨*subcaption*⟩]{⟨*text*⟩}

\subtop 명령은 subcaption 번호를 ⟨*text*⟩ 위에 놓는다. 만약 두 가지 옵션 인자가 모두 주어지면 ⟨*list-entry*⟩가 목차에 추가되고 알파벳순 식별자와 ⟨*subcaption*⟩은 ⟨*text*⟩ 위에 놓일 것이다. 한 개의 옵션 인자만 주어지면 ⟨*subcaption*⟩과 같은 것으로 취급한다. 식별자와 ⟨*subcaption*⟩은 ⟨*text*⟩ 위에 놓이고 ⟨*subcaption*⟩이 목차에 추가된다. 어떤 경우든, 식별자는 ⟨*text*⟩ 위에 오고 목차에 추가되는 것은 같다.

\subbottom 명령은 이와 동일하지만 식별자와 ⟨*subcaption*⟩이 ⟨*text*⟩ 아래에 놓인다는 점만 다르다.

메인 캡션은 플로트의 위 아래 어느쪽에든 올 수 있다.

예를 들어보자.

```
\begin{figure}
\subbottom{...} %  captioned as (a) below
\subtop{...}    %  captioned as (b) above
\caption{...}
\end{figure}
```

(a) Subfigure 1 (b) Subfigure 2

그림 11.4: Figure with two subfigures

다른 보기로, 그림 (11.4)와 그 다음 문단은 다음 코드로 만든 것이다.
그림 (11.4)는 두 개의 subfigure를 가지고 있고, 각각 11.4(a)와 (b)이다.

```
그림~\ref{subfig:sf}\는 두 개의 subfigure를 가지고 있고, 각각
\ref{sf:1}\과 \subcaptionref{sf:2}이다.
\begin{figure}
\centering
\subbottom[Subfigure 1]{\fbox{SUBFIGURE ONE}\label{sf:1}}
\hfill
\subbottom[Subfigure 2]{\fbox{SUBFIGURE TWO}\label{sf:2}}
\caption{Figure with two subfigures} \label{subfig:sf}
\end{figure}
```

\subcaption 명령과 \subtop 및 \subbottom 명령의 주요 차이점은 \subcaption이 고정폭 환경에서 이용되는 데 비해, \subtop은 ⟨*text*⟩의 폭을 이용하여 ⟨*subcaption*⟩이 식자될 박스 크기를 결정한다는 점이다.

\contsubtop[⟨*list-entry*⟩][⟨*subcaption*⟩]{⟨*text*⟩}
\contsubbottom[⟨*list-entry*⟩][⟨*subcaption*⟩]{⟨*text*⟩}

\contsubtop 명령은 (이어지는) 플로트 안에서 연속되는 sub-caption 번호를 붙이면서 ⟨*subcaption*⟩을 ⟨*text*⟩ 위에 둔다. \contsubbottom 명령은 이와 비슷하지만 ⟨*subcaption*⟩을 ⟨*text*⟩의 아래에 둔다는 점이 다르다. 어떤 경우든 플로트의 메인 캡션은 위든 아래든 붙일 수 있다.

예를 들어본다.

```
\begin{table}
\caption{...}
\subtop{...}  % captioned as (a) above
\subtop{...}  % captioned as (b) above
\end{table}
...
\begin{table}
\contcaption{Concluded}
\contsubtop{...} % captioned as (c) above
\contsubtop{...} % captioned as (d) above
```

```
\end{table}
```

\subcaption 명령에서와 같이, \label 명령은 ⟨*subcaption*⟩ 안이나, \subtop류 명령의 ⟨*text*⟩ 인자 안에 올 수 있다.

LaTeX이 캡션을 만드는 방법

이 절에서는 LaTeX이 캡션을 어떻게 만드는지 알아보고 다른 어떤 패키지도 사용하지 않고 캡션 스타일을 바꿀 수 있는 방법을 예를 들어 설명하겠다. LaTeX 코드 읽는 것을 좋아하지 않거나 LaTeX 스타일의 캡션 만들기를 변경할 생각이 없다면 굳이 읽어볼 필요는 없다.

LaTeX 커널은 캡션 정의에 필요한 도구를 제공한다. 그러나 캡션의 형태를 결정하는 것은 각각의 클래스들이다.

\caption[⟨*short*⟩]{⟨*long*⟩}

커널(ltfloat.dtx)의 캡션 정의는 다음과 같이 되어 있다.

```
\def\caption{\refstepcounter\@captype \@dblarg{\@caption\@captype}}
```

\@captype

\@captype은 새로운 플로트 환경을 만드는 코드로 정의되어 있으며, 그 환경의 이름으로 설정된다. (ltfloat.dtx의 \@xfloat 코드를 보라) figure 환경은 다음 경우에 해당한다.

```
\def\@captype{figure}
```

\@caption{⟨*type*⟩}[⟨*short*⟩]{⟨*long*⟩}

커널은 \@caption 매크로도 제공하는데, 다음과 같이 정의되어 있다.

```
\long\def\@caption#1[#2]#3{
  \par
  \addcontentsline{\csname ext@#1\endcsname}{#1}    <--
    {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}
  \begingroup
     \@parboxrestore
     \if@minipage
```

```
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par  <--
\endgroup}
```

여기서 ⟨*type*⟩은 캡션이 사용될 환경의 이름이다. 이 세 개의 명령이 함께 쓰여서 사용자 입장에서 caption 명령은 다음과 같은 형식을 띠게 된다. `\caption[`⟨*short*⟩`]{`⟨*long*⟩`}`.

이 `\ext@type`, `\fnum@type`, `\@makecaption`을 이용해서 플로트를 정의하는 것은 클래스 또는 패키지의 책임이다. 위의 예에서 <--로 표시한 행에 이것이 사용되었다.

`\ext@type`

이 매크로는 'List of...' 파일의 확장명을 기억하고 있다. `figure` 플로트 환경에서는 이 값이 다음과 같다.

```
\newcommand{\ext@figure}{lof}
```

`\fnum@type`

이 매크로는 캡션 번호를 식자하는 역할을 한다. 예를 들어 `figure` 환경이라면 이 매크로의 정의는 다음과 같다.

```
\newcommand{\fnum@figure}{\figurename~\thefigure}
```

`\@makecaption{`⟨*number*⟩`}{`⟨*text*⟩`}`

`\@makecaption` 매크로에서 ⟨*number*⟩는 'Table 5.3'과 같은 문자열이고, ⟨*text*⟩는 캡션 텍스트이다. 이 매크로는 캡션을 식자한다. 표준 클래스(`classes.dtx`)에서의 정의는 다음과 같다.

```
\newcommand{\@makecaption}[2]{
  \vskip\abovecaptionskip          <- 1
  \sbox\@tempboxa{#1: #2}         <- 2
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par                     <- 3
  \else
    \global \@minipagefalse
```

A THOUSAND WORDS...

그림 11.5: A picture is worth a thousand words

```
  \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}
\fi
\vskip\belowcaptionskip}        <- 4
```

\abovecaptionskip \belowcaptionskip

캡션의 전후에는 수직 간격이 추가된다(위의 \@makecaption 코드에서 1과 4로 표시된 부분을 보라). 각각의 길이는 \abovecaptionskip과 \belowcaptionskip이다. 표준 클래스에서는 각각 10pt와 0pt로 설정되어 있다. 캡션 전후 간격 값을 바꾸려면 \setlength를 이용한다. figure에서 캡션은 일반적으로 삽화의 아래에 놓인다. 삽화 자체의 하단과 캡션 첫 줄의 베이스라인 사이의 실제 간격은 \abovecaptionskip + \parskip + \baselineskip 이다. 삽화가 center 환경 안에 놓였다면 \end{center}에 의하여 간격이 더 추가된다. 따라서 center 환경보다는 \centering 명령을 쓰는 쪽이 더 낫다.

캡션을 실제로 식자하는 것은 결과적으로 \@makecaption 코드에서 2와 3으로 표시된 행에 나와 있는 코드에 의하여 실행된다. 여기서 번호 이후에 콜론이 식자되도록 지정되어 있음에 주의하라. 사용자 자신이 \@caption 명령을 \renewcommand하면 기본 캡션 스타일을 좀더 복잡하게 바꿀 수 있다. 그렇지만 그런 많은 변경은 \fnum@type 명령을 적절하게 수정하는 것으로도 해낼 수 있는 것이다. 예를 들어, figure 이름과 번호를 bold로 하고자 하면,

```
\renewcommand{\fnum@figure}{\textbf{\figurename~\thefigure}}
```

위와 같이 설정하는 것으로 충분하다.

REMEMBER: @ 문자가 포함된 명령을 사용하려 하고 클래스나 패키지 파일 안에서가 아니라면 \makeatletter와 \makeatother를 사용해야 한다. 따라서 \fnum@figure를 수정할 때는 문서 어디서든지 다음과 같이 전후에 이 명령들을 두어야 한다.

```
\makeatletter
\renewcommand{\fnum@figure}{......}
\makeatother
```

보기로서, 그림 (11.5)는 다음 코드로 만들어졌다.

ANOTHER THOUSAND WORDS...

그림 11.6 — A different kind of figure caption

```
\makeatletter
\renewcommand{\fnum@figure}{\textsc{\figurename~\thefigure}}
\makeatother
\begin{figure}
\centering
A THOUSAND WORDS\ldots
\caption{A picture is worth a thousand words}\label{fig:sc}
\end{figure}
```

다른 예를 하나 보자. `\figurename`과 번호를 bold 폰트로 식자하고, 번호 다음에 나오는 콜론을 긴 대시로 바꾸면서, 타이틀 텍스트는 sans-serif 폰트로 하려 한다고 하자. 이 예는 그림 (11.6)에서 본 바 있다. 다음 코드가 그것이다.

```
\makeatletter
\renewcommand{\fnum@figure}[1]{\textbf{\figurename~\thefigure}
                                --- \sffamily}
\makeatother
\begin{figure}
 \centering
 ANOTHER THOUSAND WORDS\ldots
\caption{A different kind of figure caption}\label{fig:sf}
\end{figure}
```

이것이 어떻게 이루어지는 것인지 조금 설명이 필요하겠다. TEX의 매크로 처리 과정을 조금 따라가면, `\@makecaption`의 2와 3행은 다음과 같은 형태로 변형된다.

`\fnum@figure{\figurename~\thefigure}: text`

`\fnum@figure`를 재정의할 때는 한 개의 인자를 취하는데, 그 인자 값은 사용되지 않지만 콜론 하나를 잡아먹는 효과를 가진다.

`\textbf{\figurename~\thefigure}`

위의 정의에서 `\figurename`과 번호는 bold 폰트로 식자된다. 이 처리가 끝난 후에 긴 대시가 온다. 마지막으로 `\sffamily`를 끝에 둠으로써 그 뒤에 이어지는 텍스트(즉, 타이틀)가 sans-serif 폰트로 식자되도록 하였다.

만약 `\@makecaption`을 수정한다면, 그 정의에서 스페이스는 중요하다. 그리고 위에서 보였듯이 주석처리 문자 (%)를 반드시 사용해야 한다. 이 클래스는 사용자가 필요로 하는 기능을, 물론 전부는 아니겠지만, 어떤 캡션 스타일이든 대부분 제공하기를 바라 마지 않는다.

캡션 속의 각주

만약 캡션에 각주를 붙이고 싶다면, 이것을 반드시 해야 하는지 오랫동안 깊이 생각해보기 바란다. 그것은 일반적으로 그다지 좋은 타이포그라피 관행이 아니라고 간주되고 있고 LaTeX이 쉽게 해내지 못하는 부분이기도 하다. 그렇지만, 출판사의 요구에 의해서든 저자 스스로의 판단이든 반드시 그리 해야겠다고 한다면, 이 절을 계속 읽어보라.

`\caption` 명령에 옵션 인자가 있으면 그 옵션 인자가 LoF/LoT에 들어간다. 옵션 인자가 없으면 정상적인 인자가 LoF에 들어간다. 첫번째 경우에는 옵션 인자가 가고, 두번째 경우에는 필수 인자가 간다. `\footnote` 명령은 풀리는 명령이므로 이렇게 이동하는 인자에서 사용되려면 `\protect`해주어야 하는데(즉, `\protect\footnote{}`), LoF에서 각주가 나타나야 할 이유가 없을 것이다. 그러므로 필수 인자에는 각주를 붙이고, 각주 붙이지 않은 텍스트를 옵션 인자로 주는 것이 좋다.

다음 예를 보자.

```
\begin{figure}
...
\caption[For LoF]{For figure\footnote{The footnote}}
\end{figure}
```

여기서는, (가) 각주 번호가 의도한 것보다 크다는 것, 그리고 (나) 각주 텍스트가 없어지고 나타나지 않는다는 것 때문에 당황하게 될 것이다. 두번째 일이 발생하는 이유는 LaTeX이 플로트 안에서는 각주를 식자하지 않기 때문에 발생하는 일이다. 플로트 안에서 실제로 각주를 붙이려면 minipage를 써야 한다, 다음과 같이.

```
\begin{figure}
\begin{minipage}{\linewidth}
...
\caption[For LoF]{For figure\footnote{The footnote}}
\end{minipage}
\end{figure}
```

표준 클래스를 사용하고 있다면 여기서 한 개의 각주를 지정했음에도 불구하고 두 개의 각주가 나타난다는 것을 발견했을 것이다. 다행히도 이 클래스에서는 이런 일은 벌어지지 않으며, 각주 번호가 예기치 않기 증가하지도 않는다.

위와 같이 minipage를 사용할 경우, footnote 텍스트는 minipage의 하단(즉, 플로트 내부)에 식자된다. 만약 각주를 페이지 바닥에 식자하려 한다면 `\footnotemark`와 `\footnotetext` 명령을 다음과 같이 사용해야 한다.

```
\begin{figure}
...
```

```
\caption[For LoF]{For figure\footnotemark}
\end{figure}
\footnotetext{The footnote}
```

이렇게 하면 \footnotetext 명령의 인자가 이 명령을 준 그 페이지의 바닥에 찍힌다. 물론 figure는 플로트하기 때문에, 그 뒷 페이지에 나타날 수 있다. 그래서 이런 경우에는 각주와 플로트가 같은 페이지에 있게 하려면 직접 수작업으로 위치를 찾아주어야 한다. 심지어 각주 텍스트를 각주 표지와 일치시키기 위해서 수작업을 해야할 수도 있다.

이것을 어떻게 처리할 것인가는 사용자가 직접 해결할 문제이다.

11.3 플로트

새로운 플로트 환경

\newfloat[⟨*within*⟩]{⟨*fenv*⟩}{⟨*ext*⟩}{⟨*capname*⟩}

\newfloat 명령은 ⟨*fenv*⟩와 ⟨*fenv**⟩라는 두 개의 플로트 환경을 만든다. 만약 ⟨*fenv*⟩에 해당하는 카운터가 이미 정해져 있으면 ⟨*within*⟩을 이용해서 새로 시작하게 할 수 있다. 환경 내에서 캡션은 ⟨*ext*⟩ 확장자를 갖는 파일에 쓴다. 캡션이 필요하면 ⟨*capname*⟩이라는 이름을 줄 수 있다. 예를 들어 figure 플로트는 이 클래스에서 다음과 같이 정의된 것이다.

```
\newfloat[chapter]{figure}{lof}{\figurename}
\renewcommmand{\thefigure}{%
  \ifnum\c@chapter>\z@ \thechapter.\fi \@arabic\c@figure}
```

정의의 마지막 부분은 figure가 장 번호 시작 전에 있는지를 확인하기 위한 것이다. 장이 시작하기 전이면 figure 번호에는 장 번호를 붙이지 않게 한다.

\newfloat를 이용한 플로트의 캡션 스타일은 figure나 table의 경우와 같다.

\newfloat 명령은 여러 가지 새로운 명령을 만들어낸다. 그 가운데 일부는 LaTeX 내부명령이다. 편의상 이 명령이 다음과 같은 형태로 불려졌다고 가정하자.

```
\newfloat{X}{Z}{capname}
```

여기서 X는 플로트 환경의 이름이면서 캡션 카운터의 이름이기도 하다. Z는 파일 확장자이다. 그 결과, 다음과 같은 플로트 환경과 관련 명령이 만들어진다.

\begin{X} float material \end{X}
\begin{X*} float material \end{X*}

새로운 플로트 환경은 X라는 이름을 갖는다. \begin{X}나 \begin{X*} 형태로 사용되면, \end{X}나 \end{X*}로 끝난다. 기본 위치 지정자는 tbp로 주어진다.

Zdepth

Zdepth 카운터는 tocdepth와 구조가 비슷하다. 즉, 목차에 들어간 엔트리의 레벨이 Zdepth 보다 크거나 같으면 식자되지 않는다. 기본 정의는 \setcounter{Zdepth}{1}이다. Z의 subfloat 까지 목록에 나타나게 하려면 \setcounter{Zdepth}{2}로 하여야 한다.

완전한 예제로, figure(이 클래스가 제공하는 것)와 다이어그램을 생각해보자. 다음 예제와 같이 할 수 있다.

```
\newcommand{\diagramname}{Diagram}
\newcommand{\listdiagramname}{List of Diagrams}
\newlistof{listofdiagrams}{dgm}{\listdiagramname}
\newfloat{diagram}{dgm}{\diagramname}
\newfixedcaption{\fdiagcaption}{diagram}
\newlistentry{diagram}{dgm}{0}
\begin{document}
...
\listoffigures
\listfofdiagrams
...
\begin{diagram}
\caption{A diagram} \label{diag1}
...
\end{diagram}
As diagram~\ref{diag1} shows ...
\begin{minipage}{.9\textwidth}
\fdiagcaption{Another diagram} \label{diag2}
...
\end{minipage}

In contrast to diagram~\ref{diag1}, diagram~\ref{diag2} provides ...
```

한 가지 주의를 남기자면, 플로트와 fixed 환경을 섞어서 같은 종류의 캡션을 붙이려 한다면, 최종 문서에서 순서가 정확하게 인쇄되었는지를 스스로 확인해야 한다. 만약 이렇게 하지 않으면 'List of...'에서 캡션이 나타나는 순서가 엉킬 것이다(목록은 최종 문서의 페이지 순서에 따라 정렬되고 입력 순서에 따라 정렬되지 않을 것이다).

새로운 서브플로트

subfigure 패키지는 subfigure와 subtable 서브플로트를 정의하고 있다. 이 클래스는 아무런 서브플로트도 정의하고 있지 않지만 필요하다면 정의해서 사용할 수 있다.

\newsubfloat{⟨*fenv*⟩}

\newsubfloat 명령은 \newfloat로 미리 정의된 ⟨*fenv*⟩ 플로트 환경 안에서 subcaption 명령들(\subcaption, \subtop 등)을 사용할 수 있게 한다.

\newsubfloat{fenv}를 부르면, 무엇보다도 subfenv 카운터와 \thesubfenv라는 카운터 식자 명령을 새로 만든다. \thesubfenv의 기본 정의는 아래와 같다.

```
\newcommand{\thesubfenv}{(\alph{subfenv})}
```

이 명령은 괄호로 둘러싸인 알파벳 소문자로 식자한다. 이것이 서브캡션의 식별자이다. \renewcommand 명령으로 다른 모양으로 바꿀 수 있다.

만약 subfigure 패키지를 사용하고, 예컨대 subfigure를 쓰고자 한다면, preamble에서 다음과 같이 하라.

```
\newsubfloat{figure}
```

그리고 subcaption이 List of Figures에 나타나게 하려면, 다음 코드를 \begin{document}와 \listoffigures 명령 사이에 둔다.

```
\setcounter{lofdepth}{2}
```

만약 subfigure를 원래 subfigure 패키지를 이용해서 사용하려고 만들었는데 이 패키지 없이 쓰고 싶다면,

```
\let\subfigure\subbottom
```

이렇게 하면 \subfigure가 \subbottom의 역할을 하게 된다.

다중 플로트

LaTeX에서 플로트란 놓일 수 있는 위치가 제한되는 박스이다. 결국 플로트 박스 안에는 무엇이든 넣을 수 있다. 일반적으로 하나의 그림이나 표가 오지만 박스 안에 여러 개의 그림이나 표를 두는 것도 가능하다.

하나의 플로트 안에 여러 개의 그림이나 표가 온다면 생각해볼 수 있는 것은 다음 세 가지 경우이다.

ILLUSTRATION 1 ILLUSTRATION 2

그림 11.7 — Float with two illustrations

- 여러 개의 그림이나 표를 넣고 캡션은 하나를 붙인다.
- 여러 개의 표나 그림을 넣고 각각 캡션을 붙인다.
- 여러 개의 표나 그림을 넣고 각각 subcaption을 붙이면서 전체에 대해서 메인 캡션을 붙인다.

subfigure 패키지가 처리하는 것은 이 가운데 세번째 경우이다. 다른 경우에는 패키지가 필요하지 않다.

그림 (11.7)은 여러 개 그림을 하나의 플로트에 넣고 하나의 캡션을 붙인 예이다. 이 그림은 다음 코드로 작성되었다.

```
\begin{figure}
\centering
\hspace*{\fill}
  {ILLUSTRATION 1} \hfill {ILLUSTRATION 2}
\hspace*{\fill}
\caption{Float with two illustrations} \label{fig:mult1}
\end{figure}
```

\hspace*{\fill}과 \hfill 명령을 사용하여 두 그림에 동일한 간격을 주었다. 물론 이 예제의 {ILLUSTRATION N} 텍스트 대신 \includegraphics나 tabular 환경을 넣어서 마찬가지로 할 수 있다.

다음 코드는 그림s 11.8과 11.9를 만드는 것이다. 여기서는 하나의 플로트에 각각 별도의 캡션이 붙은 그림을 넣고 있다.

```
\begin{figure}
\centering
\begin{minipage}{0.4\textwidth}
  \centering
  ILLUSTRATION 3
  \caption{Illustration 3} \label{fig:mult2}
\end{minipage}
\hfill
\begin{minipage}{0.4\textwidth}
  \centering
  ILLUSTRATION 4
  \caption{Illustration 4} \label{fig:mult3}
\end{minipage}
\end{figure}
```

ILLUSTRATION 3

그림 11.8 — Illustration 3

ILLUSTRATION 4

그림 11.9 — Illustration 4

이 경우, 그림 (또는 graphics나 tabulars)은 별도의 `minipage` 환경에 들어가서 플로트 안에 놓여야 한다. 그리고 캡션 역시 `minipage` 안에서 붙여야 한다. `\label` 역시 `minipage` 안에 두어야 한다. 필요하다면 `minipage` 두 개를 넣은 다음에 또다른 캡션을 둘 수도 있다.

Keith Reckdahl [Rec97]은 이런 예를 몇 가지 더 들고 있다.

LaTeX이 플로트를 놓는 위치

플로트 환경의 일반적인 형식은,

`\begin{float}[`⟨*loc*⟩`] ... \end{float}`

또는, 2단 플로트

`\begin{float*}[`⟨*loc*⟩`] ... \end{float*}`

여기서 ⟨*loc*⟩이라는 옵션 인자는 하나 이상의 문자로 구성되는 것으로서 플로트가 놓일 위치를 지정한다. multicol 패키지 [Mit98]는 오직 별표붙은 플로트만을 지원하고 한 단에만 가두어지는 플로트를 사용할 수 없게 한다는 점을 기억하자. ⟨*loc*⟩에 올 수 있는 값은 다음 중의 하나이거나 둘 이상이다.

- `b` *bottom*: 페이지의 바닥에 놓는다. 이 옵션은 2단 플로트를 지원하지 않는다. 페이지의 위쪽에 플로트를 놓을 것이다.
- `h` *here*: 될 수 있는 대로 플로트 환경이 정의된 위치에 놓으려 한다. 2단 플로트는 지원하지 않는다.
- `p` *page*: 별도의 플로트만을 두는 페이지를 만든다.
- `t` *top*: 페이지의 상단에 둔다.
- `!` 플로트를 이 뒤에 오는 인자가 지시하는 위치 중에서 제일 첫번째 것에 해당하는 위치에 두려고 노력한다.

⟨*loc*⟩의 기본값은 `tbp`이다. 따라서 플로트는 상단, 바닥, 또는 플로트만의 페이지에 위치한다. 이 기본값은 약 95% 정도로 플로트를 잘 처리한다. 같은 종류의 플로트는 정의된 순서대로 나타난다. 다만, 2단 플로트는 그 뒤에 나오는 1단 플로트보다 먼저 나오거나 뒤에 나타날 수 있다.[2] 플로트는 그것이 정의된 페이지보다 앞쪽으로 가는 경우는 없다. 대체로 정의된 그 페이지나 그 뒷페이지에 나타난다. 플로드를 두는 데 실패하면, 그 이후의 모든 플로트를 모두 처리하지 못한다. 그리고 LaTeX이 저장할 수 있는 플로트의 갯수는

[2] 이 약간의 문제점은 fixltx2e 패키지로 수정할 수 있다. 최소한 표와 그림에 대해서는 잘 된다. 이 패키지는 기본 LaTeX 배포판의 일부이다.

16개를 넘지 않는다. 플로트를 두는 데 실패한다는 것은, overfull 페이지가 되거나, 플로트 파라미터에 맞출 수가 없다는 것을 의미한다. `\clearpage` 또는 `\cleardoublepage`, `\end{document}` 이 세 명령은 그 시점까지 처리되지 않은 플로트를 ⟨*loc*⟩나 플로트 파라미터를 무시하고 플로트 페이지를 만들어서 모두 식자한다.

`\suppressfloats[`⟨*pos*⟩`]`

`\suppressfloats` 명령은 현재 페이지의 ⟨*pos*⟩ 위치에 플로트가 오는 것을 제한할 수 있다. `\suppressfloats[t]`는 현재 페이지의 top 위치에 어떤 플로트도 오지 못하게 한다. 그리고 `\suppressfloats[b]`는 현재 페이지의 bottom 위치에 플로트가 오지 못하게 한다. 단순히 `\suppressfloats`라고 하면 top과 bottom 플로트를 금지하는 것이다.

flafter 패키지는 LaTeX 기본 배포판에 포함되어 있는 것으로서, 텍스트 상 정의되는 위치 이전으로 플로트가 가지 못하게 하는 역할을 한다. 이것은 예를 들면 `\section{}` 명령보다 플로트가 먼저 나와서 절 헤딩보다도 플로트가 먼저 식자되는 것을 막을 수 있게 한다.

그림 (11.10)과 11.11은 여러 가지 플로트 설정변수를 묘사하고 있다. 그리고 표 [11.4]는 플로트 설정변수와 그 기본값을 보여준다. 모든 길이는 가변길이변수이고 실제 기본값은 클래스와 사이즈 옵션에 따라 달라진다.

기본값에 따르면 top 플로트는 textheight의 70%보다 작아야 하고, 한 페이지에는 top 플로트가 두 개 이상 올 수 없다. bottom 플로트의 height는 textheight의 30%를 넘어야 하고 한 페이지에 하나 이상의 플로트가 올 수 없다. 한 페이지에는 세 개(top, bottom, here) 이상의 플로트가 올 수 없다. 텍스트 페이지에서는 최소한 20%는 텍스트여야 한다. 플로트 페이지(텍스트 없이 플로트만 들어가는 페이지)에서 float의 height의 합계는 최소한 textheight의 50%는 되어야 한다. 플로트 페이지에서 플로트의 위치는 수직으로 가운데이다.

LaTeX 기본값을 적용하면 플로트를 어디에 두어야 할지 알 수 없는 경우가 있다. 예를 들어 textheight의 40% 높이를 가진 bottom 플로트가 textheight의 90% 높이를 가진 플로트 뒤에 오게 되는 경우를 생각해보자. 첫번째 것은 텍스트 페이지의 bottom에 오기에는 너무 크고 그것만으로 플로트 페이지를 만들기에는 너무 작다. 두번째 것은 플로트 페이지로 가야할 것이지만 앞의 플로트와 함께 플로트 페이지를 만들기에는 너무 크다. LaTeX은 어찌할 수 없게 된다.

여기서 잠깐, ⟨*loc*⟩ 인자가 가져오는 효과를 요약해보자.

`[b]` '플로트를 텍스트 아래 바닥에 둘 것. 다른 곳은 안된다.'는 뜻이다. 플로트가 `\bottomfraction` 스페이스에 맞아들어가야 한다. 그렇지 않으면 그 뒤의 플로트들까지 처리하지 못한다.

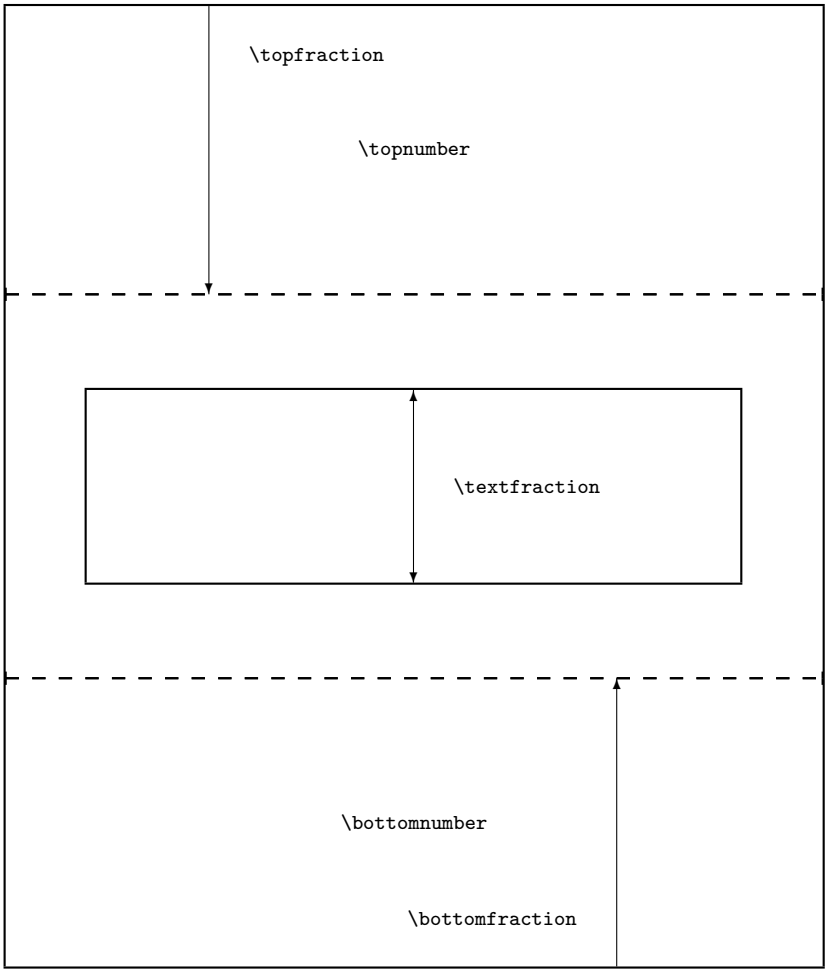


그림 11.10 — Float and text page parameters

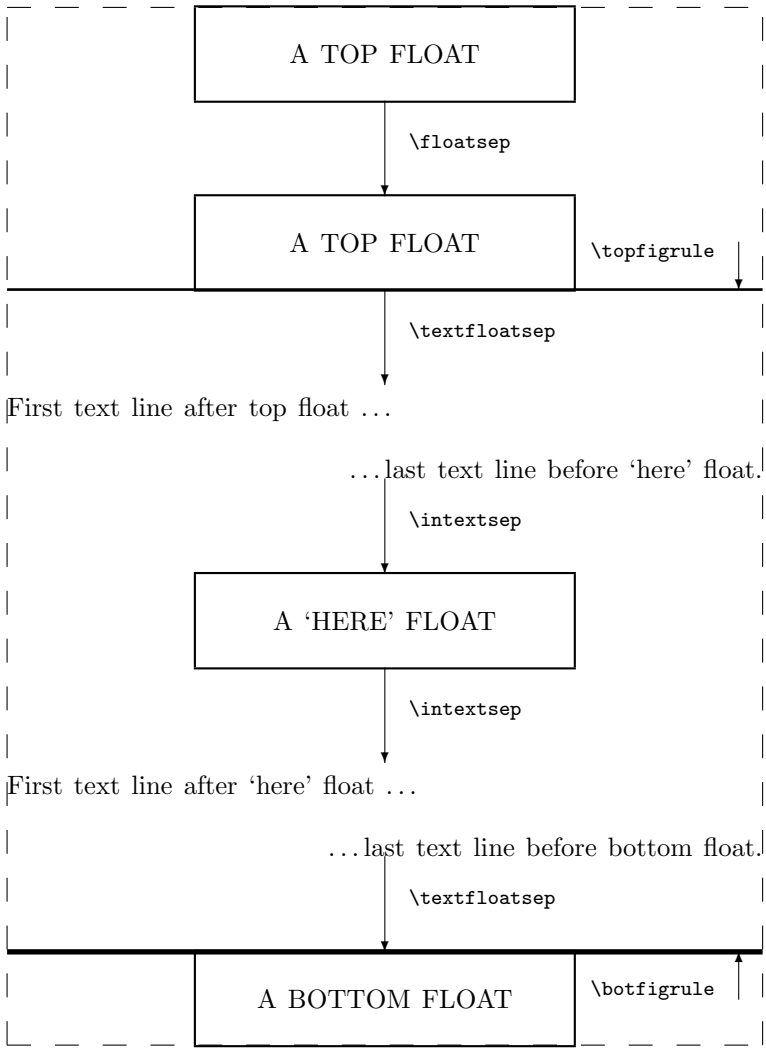


그림 11.11 — Float parameters

표 11.4: Float placement parameters

Parameter	Controls	Default
	Counters — change with `\setcounter`	
`topnumber`	max number of floats at top of a page	2
`bottomnumber`	max number of floats at bottom of a page	1
`totalnumber`	max number of floats on a text page	3
`dbltopnumber`	like `topnumber` for double column floats	2
	Commands — change with `\renewcommand`	
`\topfraction`	max fraction of page reserved for top floats	0.7
`\bottomfraction`	max fraction of page reserved for bottom floats	0.3
`\textfraction`	min fraction of page that must have text	0.2
`\dbltopfraction`	like `\topfraction` for double column floats	0.7
`\floatpagefraction`	min fraction of a float page that must have float(s)	0.5
`\dblfloatpagefraction`	like `\floatpagefraction` for double column floats	0.5
	Text page lengths — change with `\setlength`	
`\floatsep`	vertical space between floats	12pt
`\textfloatsep`	vertical space between a top (bottom) float and suceeding (preceding) text	20pt
`\intextsep`	vertical space above and below an `h` float	12pt
`\dblfloatsep`	like `\floatsep` for double column floats	12pt
`\dbltextfloatsep`	like `\textfloatsep` for double column floats	20pt
	Float page lengths — change with `\setlength`	
`\@fptop`	space at the top of the page	`0pt plus 1fil`
`\@fpsep`	space between floats	`8pt plus 2fil`
`\@fpbot`	space at the bottom of the page	`0pt plus 1fil`
`\@dblfptop`	like `\@fptop` for double column floats	`0pt plus 1fil`
`\@dblfpsep`	like `\@fpsep` for double column floats	`8pt plus 2fil`
`\@dblfpbot`	like `\@fpbot` for double column floats	`0pt plus 1fil`

`[h]` '플로트를 이곳에 둘 것. 다른 곳은 안된다'는 뜻이다. 플로트는 그 페이지의 남은 공간에 맞아들어가야 한다. 그렇지 않으면 이 플로트와 그 뒤의 플로트가 처리되지 못한다.

`[p]` '플로트를 텍스트가 놓이지 않는 별도의 페이지에 둘 것. 다른 플로트와 함께 두어도 좋다.'는 뜻이다. 최소한 `\floatpagefraction` 값이 차야 그 때까지 모인 플로트를 모아서 하나의 페이지로 만든다.

`[t]` '플로트를 페이지의 상단에 두고 텍스트는 그 아래로 흐르게 할 것. 다른 곳은 안된다.'는 뜻이다. 플로트는 `\topfraction` 공간에 맞아들어가야 한다. 그렇지 않으면 그것과 그 이후의 플로트가 처리되지 못한다.

`[!...]` 이것은 `\...fraction` 값을 이 플로트에 적용하지 말라는 뜻이다.

이 옵션들을 하나씩 그리고 결합해서 시험해보면서 LaTeX이 적절한 플로트 놓을 위치를 찾도록 시도해보는 방법이 있다. 그러나, 좀더 쉽게 플로트를 둘 위치를 찾도록 플로트 파라미터들을 변경할 수도 있다. 몇 가지 보기를 들면,

- `\textfraction` 값을 줄여서 텍스트 페이지에 더 많은 '플로트'를 두게 할 수 있다. 그러나 `\textfraction`과 `\topfraction`을 합한 값과 `\textfraction`과 `\bottomfraction`을 합한 값이 1.0을 넘어서는 안된다. `\textfraction`의 최소값은 0.10 정도이다. 10% 미만의 텍스트를 가진 페이지는 차라리 텍스트 없이 플로트만 두는 것이 낫다.
- `\topfraction`과 `\bottomfraction`은 둘 다 값을 증가시킬 수 있다. 그러나 그 합이 1.0을 넘어서는 안된다. 일반적으로 플로트는 페이지의 상단에 두는 것이 권장되고 있고, 상단 플로트 영역이 하단 플로트 영역보다 넓은 쪽이 타이포그라피상 더 낫다고 알려져 있다.
- `\floatpagefraction`을 너무 작게 하면 플로트 페이지에 작은 플로트 하나만 오게 되는 경우가 생긴다. 그러나 플로트 페이지가 하나의 플로트 만을 갖지 않도록 하려면, 다음과 같이 한다.

```
\renewcommand{\floatpagefraction}{0.01}
\setlength{\@fpsep}{\textheight}
```

- `\@fptop` 길이를 `0pt`로 하고, `\@fpsep` 길이를 `8pt`로 하고, `\@fpbot` 길이를 `0pt plus 1fil`로 강제하면 플로트 페이지에서 플로트가 페이지 상단에 놓일 것이다.

만약 시험해볼 생각이라면, 다음 값에서부터 출발하는 것이 좋다.

```
\setcounter{topnumber}{3}
\setcounter{bottomnumber}{2}
\setcounter{totalnumber}{4}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.5}
\renewcommand{\textfraction}{0.15}
\renewcommand{\floatpagefraction}{0.7}
```

그리고 2단 플로트의 경우에도 이와 비슷하게 할 수 있다. 사실은, 이들 설정값을 테스트해볼 필요는 별로 없는데, 바로 이 클래스의 기본값이기 때문이다.

LaTeX의 특이한 점 중의 하나는, 텍스트 페이지에서 플로트의 'height'라는 것은 실제 높이와 `\textfloatsep` 또는 `\floatsep`을 합한 값이라는 점이다. 그러나 플로트 페이지에서 'height'는 실제 높이만을 의미한다. 그러므로, ⟨*loc*⟩ 기본값 `[tbp]`를 사용한다면 텍스트 페이지의 플로트 비율(`\topfraction` 또는 `\bottomfraction`) 가운데 하나는 `\floatpagefraction`보다 텍스트 페이지 구분 최대값을 취하기에 충분할 정도의 값만큼 커야 한다.

12 행과 열

12.1 들어가는 말

이 클래스는 표준 array와 tabular 환경을 확장하였다. 부분적으로 다음 스타일 패키지들을 병합하여 구현하였다. array [MC98], dcolumn [Car01], delarray [Car94], tabularx [Car99a],booktabs [Fea03]. 이 장에서 사용된 대부분의 내용은 이 패키지의 문서에서 가져온 것이 많다.

그러나, tabular 환경을 확장한 것은 새로운 것이다.

12.2 개요

\[\begin{array}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{array} \]
\begin{tabular}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{tabular}
\begin{tabular*}{⟨*width*⟩}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{tabular*}
\begin{tabularx}{⟨*width*⟩}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{tabularx}

array와 tabular 환경이 기본적인 것이고 나머지는 그것을 확장한 것이다. array는 수학식 조판에 쓰이며, 수학식 환경 안에 있어야 한다. tabular 종류는 일반적인 텍스트를 조판할 때 사용된다.

⟨*pos*⟩ 옵션 인자는 t, c, b 가운데 하나인데, 디폴트는 c이다. 이것은 array나 tabular의 세로 정렬 방식을 제어한다. 각각 top, center, bottom을 표시하고, 이름 그대로 베이스라인을 기준으로 정렬된다. 각 행 구성요소는 & 문자로 분리되며, \\로 행 끝임을 표시한다. 행은 얼마든지 둘 수 있다. 열(column)의 개수와 형식은 ⟨*preamble*⟩에 의하여 지정된다. 각 컬럼의 폭은 가장 긴 엔트리가 들어갈 수 있을 정도로 잡힌다. array, tabular의 전체 가로 폭은 모든 컬럼의 폭을 모두 포함하는 정도의 길이가 된다. 그렇지만 tabular*와 tabularx는 ⟨*width*⟩ 인자로 이 폭의 길이를 제어할 수 있다.

표 12.1: array와 tabular의 설정부 옵션.

l	왼쪽 정렬된 컬럼
c	중앙정렬된 컬럼
r	오른쪽 정렬된 컬럼
p{⟨*width*⟩}	\parbox[t]{⟨*width*⟩}와 동일함.
m{⟨*width*⟩}	컬럼 폭을 ⟨*width*⟩로 설정. 모든 엔트리는 남은 행이 얼마냐에 따라서 (세로로) 중앙 위치에 놓임. \parbox{⟨*width*⟩}와 비슷함.
b{⟨*width*⟩}	\parbox[b]{⟨*width*⟩}에 해당함.
>{⟨*decl*⟩}	l, r, c, p, m, b 옵션 앞에 놓여서 ⟨*decl*⟩을 컬럼 엔트리 식자 전에 직접 삽입함.
<{⟨*decl*⟩}	l, r, c, p{..}, m{..}, b{..} 옵션 뒤에 놓여서 컬럼 엔트리 식자 이후에 ⟨*decl*⟩을 직접 삽입함.
|	수직선을 긋는다. 두 컬럼 사이의 간격은 선의 두께 만큼 늘어남.
@{⟨*decl*⟩}	컬럼 간 간격을 없애고 ⟨*decl*⟩을 그 대신 삽입한다.
!{⟨*decl*⟩}	|를 쓸 곳에 사용할 수 있다. 차이점은 ⟨*decl*⟩이 수직선 대신 들어간다는 것인데, @{...}를 쓰는 경우와는 달리 컬럼 간 간격을 없애지 않는다.
D{⟨*ssep*⟩}{⟨*osep*⟩}{⟨*places*⟩}	컬럼 엔트리가 '소수점'을 기준으로 정렬된다.

12.3 설정부(preamble)

array와 tabular 환경에서, ⟨*preamble*⟩ 인자를 사용하여 컬럼의 개수와 컬럼 엔트리 정렬 방식을 지시한다. 설정부는 표 [12.1]에 보인 것과 같은 옵션으로 이루어진다.

위의 옵션이 사용된 예를 들어보자.

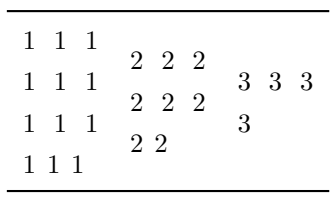
- >{\bfseries}l을 지정하여 컬럼 내용을 굵은 글꼴의 flush left로 만든다.

```
\begin{center}
\begin{tabular}{>{\large}c >{\large\bfseries}l >{\large\itshape}c} \toprule
A   & B  & C \\
100 & 10 & 1 \\ \bottomrule
\end{tabular}
\end{center}
```

A	**B**	*C*
100	**10**	*1*

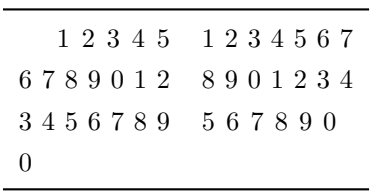
- p, m, b로 만들어지는 컬럼을 비교한다. \parindent 기본값은 0pt이다.

```
\begin{center}
\begin{tabular}{m{1cm}m{1cm}m{1cm}} \toprule
1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2         &
3 3 3 3                 &
\\ \bottomrule
\end{tabular}
\end{center}
```



\parindent는 예컨대 >{\setlength{\parindent}{1cm}}p와 같은 방식으로 바꿀 수 있다.

```
\begin{center}
\begin{tabular}{>{\setlength{\parindent}{5mm}}p{2cm} p{2cm}} \toprule
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 &
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 \\ \bottomrule
\end{tabular}
\end{center}
```



- >{$}c<{$}는 tabular 환경 내에서 수학 모드의 컬럼을 만들어준다. array 환경 안에서는 오히려 LR 텍스트 모드로 바뀌는데, 그 이유는 첫번째 $가 기존의 수학 모드에서 텍스트 모드로 shift해주는 결과를 가져오기 때문이다.
- c!{\hspace{1cm}}c를 사용하여 두 컬럼 사이의 간격을 설정할 수 있다. 여기서는 1센티미터를 더 늘려보았다. 반면, c@{\hspace{1cm}}c로 하면 컬럼 사이의 간격이 정확하게 1센티미터가 된다.
- 다른 곳에서 표에 수직선(즉, |)을 사용하지 않는 것이 좋은 이유를 설명하였다. 수직선이 사용된 예제는 컬럼 경계를 보여주고 간격 효과를 확인하기 위한 목적일 뿐이다.

컬럼 지시자 D

재무관련 표에서는 파운드, 펜스, 달러, 센트 등을 취급하게 되는데, 이 때 컬럼 엔트리가 숫자 사이의 구분자에 따라 정렬되어야 한다. D 컬럼 지시자는 컬럼을 '소숫점'에 따라 정렬한다. 이 지시자는 세 개의 인자를 취한다.

D{⟨*ssep*⟩}{⟨*osep*⟩}{⟨*places*⟩}

⟨*ssep*⟩ 이 인자는 .tex 파일 소스에서 구분자로 사용된 문자 하나를 가리킨다. 일반적으로 '.'이거나 ','이다.

⟨*osep*⟩ 이 인자는 출력 결과에서 사용될 구분자이다. 첫번째 인자와 같을 수 있지만, \cdot 과 같은 수학식 표현이 올 수도 있다. D 컬럼 지시자는 숫자나 구분자 모두 언제나 수학식 모드로 식자한다.

⟨*places*⟩ 이 인자는 컬럼의 소수부(좀더 자세한 것은 아래를 참고하라)의 최대 자리수를 지시한다. 이 값이 음수이면 소수 자리수에 제한이 없고 소숫점은 컬럼의 중앙에 놓인다. 이 경우 컬럼이 너무 넓어질 수 있다는 데 주의하라. 아래 예 중에서 처음 두 개의 컬럼을 비교해보면 알 수 있을 것이다.

여기서 몇 가지 예를 들어볼 것인데, 편의상 \newcolumntype 매크로를 사용하겠다. 이 매크로에 대해서는 나중에 설명한다.

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
```

이것은 d라는 컬럼 지시자를 정의하여 한 개의 인자를 취한다. 이 인자는 소수점의 위치를 표시한다. .tex 파일에는 소수점 분리자가 '.'으로 쓰여야 하고, \cdot (·)이 출력에 사용된다.

```
\newcolumntype{.}{D{.}{.}{-1}}
```

위와 같이 한 결과는 '.' 컬럼 지시자를 정의하여 엔트리가 가운데 놓인 '.'를 기준으로 정렬되게 하는 것이다.

```
\newcolumntype{,}{D{,}{,}{2}}
```

위의 결과는 ',' 컬럼 지시자를 정의하여 컬럼 엔트리를 소수점 이하 최대 두 자리까지만 ',' 뒤에 표시하도록 하는 것이다.

다음 표는 아래 코드로 작성되었다.

```
\begin{center}
 \begin{tabular}{|d{-1}|d{2}|.|,|}
 1.2   & 1.2   &1.2    &1,2    \\
 1.23  & 1.23  &12.5   &300,2  \\
 1121.2& 1121.2&861.20 &674,29 \\
 184   & 184   &10     &69     \\
 .4    & .4    &       &,4     \\
       &       &.4     &
```

```
\end{tabular}
\end{center}
```

1·2	1·2	1.2	1,2
1·23	1·23	12.5	300,2
1121·2	1121·2	861.20	674,29
184	184	10	69
·4	·4		,4
		.4	

첫번째 컬럼에 ⟨*places*⟩ 인자로 음수값을 주었는데, 그 결과 두번째 컬럼보다 폭이 넓어졌다. 그리고 소수점이 컬럼의 중앙에 오고 있다.

⟨*places*⟩ 세번째 인자는 소수점을 기준으로 왼쪽과 오른쪽 숫자를 모두 지정한다. 다음 표의 세번째 컬럼은 첫번째 표에서는 `D{.}{.}{5.1}`로 정의되었고, 두번째 표에서는 `D{.}{.}{1.1}`로 지정하였는데, 각각 소수점 기준으로 '왼쪽으로 다섯 자리, 오른쪽으로 한 자리'와 '왼쪽으로 한 자리, 오른쪽으로 한 자리'를 의미한다. (이 인자에서는 굳이 '.'를 쓰지 않고 필요하다면 ','나 다른 문자를 써도 상관없다.) 실제 표현된 표에서 지정된 자릿수만큼의 숫자가 가운데 오도록 정렬된다.

만약 원한다면 테이블 헤딩을 넣을 때, (`\multicolumn{1}{c}{...}`를 삽입하는 방법으로) `D` 컬럼 타입을 덮어쓸(override) 수 있다. 이럴 경우에는 '중앙정렬' 또는 '우측정렬' 형식도 더이상 작동하지 않는다.

```
 \begin{center}\small
 \begin{tabular}[t]{|D..{-1}|D..{1}|D..{5.1}|}
\multicolumn{1}{|c|}{head}&
\multicolumn{1}{c|}{head}&
\multicolumn{1}{c|}{head}\\[3pt]
 1.2  & 1.2  &1.2 \\
 11212.2& 11212.2&11212.2  \\
 .4    & .4    &.4
 \end{tabular}
 \hfill
 \begin{tabular}[t]{|D..{-1}|D..{1}|D..{1.1}|}
\multicolumn{1}{|c|}{wide heading}&
\multicolumn{1}{c|}{wide heading}&
\multicolumn{1}{c|}{wide heading}\\[3pt]
 1.2  & 1.2  &1.2 \\
 .4    & .4    &.4
 \end{tabular}
 \end{center}
```

head	head	head
1.2	1.2	1.2
11212.2	11212.2	11212.2
.4	.4	.4

wide heading	wide heading	wide heading
1.2	1.2	1.2
.4	.4	.4

이 두 개의 표는 모두 첫번째 컬럼이 `D{.}{.}{-1}`로 지시되어 있고, '.'을 컬럼의 중앙에 두도록 하고 있다. 두번째 컬럼은 `D{.}{.}{1}`로 설정하여서 오른쪽 정렬하였다.

중앙정렬된 (첫번째 표) 컬럼은 숫자가 들어가기에 필요한 폭보다 컬럼 폭이 넓어지게 되었는데, 그것은 소수점을 중앙에 두도록 했기 때문이다. 오른쪽 정렬된 (두번째 표) 컬럼은 이러한 여분 공백을 가지고 있지 않지만 작은 숫자가 오른쪽으로 정렬된 반면 헤딩이 너무 넓어서 부조화스럽다.

⟨*places*⟩ 인자 지시를 이용하여, 숫자의 시작점이 아니라 중앙에 놓인 구분자를 기준으로 컬럼을 배열하게 할 수 있다. 예를 들면 `D{+}{\,\pm\,}{3,3}`으로 하는 경우 ± 부호를 기준으로 양쪽에 세 자리씩의 숫자가 대칭적으로 오도록 해준다.

새로운 컬럼 지시자의 정의

간단히 다음과 같이 입력하면, 표의 한 컬럼 모양을 지정할 수 있다.

`>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}`

그러나 이런 형식을 자주 사용해야 하는데 매번 이와 같이 입력하는 것은 피곤한 일이므로, `\newcolumntype` 명령을 써서 새로운 컬럼 옵션을 정의할 수 있다. 즉,

`\newcolumntype{x}{>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}}`

이렇게 함으로써 `x`라는 컬럼 타입을 언제라도 이런 형식의 컬럼이 필요할 때 설정부에 지정할 수 있는 것이다.

`\newcolumntype{`⟨*char*⟩`}[`⟨*nargs*⟩`]{`⟨*spec*⟩`}`

⟨*char*⟩ 인자는 컬럼 지시자 명칭으로 쓰이는 문자이고, ⟨*spec*⟩은 표준 설정부 표현으로 된 설정 명령 부분이다. `\newcolumntype` 명령은 `\newcommand` 명령과 비슷하다. ⟨*spec*⟩ 그 자체는 ⟨*nargs*⟩로 지정된 숫자 만큼의 인자를 취할 수 있다.

예를 들면, 수학식 컬럼과 텍스트 컬럼이 동일하게 정렬해야 할 때가 있다.

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}
```

C는 array에서는 가운데 정렬된 텍스트 모드가 되고, tabular에서는 가운데 정렬된 수학모드가 된다. L과 R에 대해서도 마찬가지로 왼쪽정렬 또는 오른쪽정렬된 컬럼이 된다.

\newcolumntype의 ⟨*spec*⟩은 기본 컬럼 지시자(192 쪽의 표 [12.1]을 보라)일 수도 있고, 다른 \newcolumntype 명령으로 만들어진 새로운 컬럼 지시문자일 수도 있다.

```
\showcols
```

\showcols 명령을 쓰면 현재 활성화된 \newcolumntype 정의들의 목록은 터미널에 표시되고 log 파일에 기록된다.

주의

- {wx*{0}{abc}yz} 형식의 설정부는 {wxyz}로 취급된다.
- 예를 들어 [Q]와 같은 잘못된 위치 지정 인자는 [t]로 취급된다.
- 예컨대 {cc*{2}}와 같은 잘못된 설정부는 ∗-사용형식에서 에러를 낸다.
- 컬럼 지시자를 검토하는 과정에서 발생하는 에러 메시지는 사용자가 설정부를 입력한 곳을 가리키지 않고 \newcolumntype 시스템이 재정의된 0이후의 설정부 인자를 가리킨다.
- <나 > 구문은 반복 사용할 수 있다. >{⟨*decs1*⟩}>{⟨*decs2*⟩}는 >{⟨*decs2*⟩}{⟨*decs1*⟩}과 같이 취급된다.

  중복사용한 <나 > 선언을 취급하는 방법이 약간 이상해보일 수 있는데, 명시적으로 >{⟨*decs1*⟩}{⟨*decs2*⟩}와 같이 취급하게 되면 이 선언이 \newcolumntype 설정을 오버라이드하게 할 수가 없다는 문제점이 있어서 이러한 방식으로 처리된다.
- \extracolsep 명령은 LaTeX에서와 마찬가지로 @-표현과 함께 쓰일 수 있다. 또한, !-표현과도 함께 쓸 수 있다.

  \extracolsep의 사용은 두가지 제한이 있다. 적어도 \extracolsep 명령이 @나 ! 표현에서 하나는 사용되어야 한다. 그리고 이 명령은 다른 매크로 정의의 일부로 사용되어서는 안되고 @ 표현에 직접 삽입되어야 한다. 따라서,

  ```
  \newcommand{\ef}{\extracolsep{\fill}} ... @{\ef}
  ```

  위의 코드는 작동하지 않을 것이다. 제대로 하려면 다음과 같이 하여야 한다.

  ```
  \newcolumntype{e}{@{\extracolsep{\fill}}
  ```
- LaTeX 교재에서 언급하고 있는 대로, \multicolumn은, 첫번째 컬럼의 경우를 예외로 하고, 엔트리와, 그 0이후의 컬럼 사이에 올 내용으로 이루어져 있다. 이 말은 설정부가 |l|l|l|l|로 된 tabular에서 \multicolumn{2}{|c|}와 같이 입력하는 것은 첫번째 컬럼이 아니라면 잘못이라는 의미이다.

  표준 array/tabular 구현에서 이 에러가 눈에 띄지 않는 이유는 |이 폭을 갖지 않기 때문이다. 그러나 이 클래스에서 수직선은 그 자신의 폭을 가지는 것이므로, 두 개의 수직선으로 나타나게 된다. | 사용을 피해야 하는 또하나의 이유이다.

12.4 array 환경

배열 형태의 수학식은 array 환경으로 만들어진다.

\[\begin{array}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{array} \]
\[\begin{array}[⟨*pos*⟩]⟨*left*⟩{⟨*preamble*⟩}⟨*right*⟩ rows \end{array} \]

수학식은 대부분 컬럼에서 가운데정렬된다. 그렇지만 숫자로 된 컬럼은 flush right되거나 특정한 위치를 기준으로 정렬되었을 때 보기좋을 수도 있다. 후자의 경우 D 컬럼 정렬방식이 편리하다.

```
\[ \begin{array}{lcr}
   a +b +c & d - e - f & 123 \\
   g-h     &  j k      & 45 \\
    l      &   m       & 6
  \end{array} \]
```

$$\begin{array}{lcr} a+b+c & d-e-f & 123 \\ g-h & jk & 45 \\ l & m & 6 \end{array}$$

배열은 대괄호나 수직선, 또는 기타 기호로 둘러싸서 행렬과 같은 형태를 만들기도 한다. 경계의 짝맞춤 문자는 큰 글자이고 \left나 \right 명령을 이용하여 지시된다.

```
\[ \left[ \begin{array}{cc}
          x_{1} & x_{2} \\
          x_{3} & x_{4}
          \end{array} \right] \]
```

$$\left[\begin{array}{cc} x_1 & x_2 \\ x_3 & x_4 \end{array} \right]$$

array 환경은 표준 환경의 확장으로서, \left \right 쌍이 있을 것을 예상하고 있다. array를 괄호로 둘러싸려면 다음과 같이 입력하라.

```
\[  \begin{array}({cc})a&b\\c&d\end{array}   \]
```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right)$$

또는 조금 복잡하게 다음처럼 할 수도 있다.

```
\[ \begin{array}({c})
     \begin{array}|{cc}|
       x_{1} & x_{2} \\
       x_{3} & x_{4}
     \end{array} \\
        y \\
        z
   \end{array} \]
```

$$\left(\begin{array}{|cc|} x_1 & x_2 \\ x_3 & x_4 \end{array} \right) \\ y \\ z$$

마찬가지로, plain TEX의 \cases에 해당하는 환경은 다음과 같이 정의할 수 있을 것이다.

```
\[  f(x)=\begin{array}\{{lL}.
          0        &if $x=0$\\
          \sin(x)/x&otherwise
          \end{array}  \]
```

$$f(x) = \left\{ \begin{array}{ll} 0 & \text{if } x = 0 \\ \sin(x)/x & \text{otherwise} \end{array} \right.$$

여기서 L은 왼쪽정렬된 L-R 텍스트임을 의미한다. 짝맞춤 문자는 반드시 쌍으로 쓰여야 한다는 점을 명심하자. 한쪽을 표시하지 않으려면 '.'를 사용한다.

이것은 특히 [t]나 [b] 인자가 사용될 때 유용하다. 이 경우에 결과는 \left...\right 환경으로 둘러싼 것과는 다르게 나타난다. 다음 보기를 보자.

```
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 1\\2\\3 \end{array}
\begin{array}[b]({c}) 1\\2\\3 \end{array}
\quad\mbox{not}\quad
\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{not} \quad \begin{pmatrix} \\ 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ \end{pmatrix}$$

12.5 표(Tables)

테이블은 제한된 공간에 많은 정보를 전달하는 방법이다. 단순한 표라 하더라도 상당한 양의 말로써 설명해야 할 것을 나타내고 있다. 즉, 그것은 단어로 이루어진 그림에 불과한 것이 아니다.

테이블에는 최소한 두 개 이상의 컬럼이 있어야 한다. 그렇지 않다면 그것은 나열 문단(list)일 뿐이다. 가장 왼쪽 컬럼은 스터브(stub)라 불리고, 다른 컬럼에 포함된 내용에 대한 설명 정보가 세로로 나열된다. 컬럼은 난제(欄題, column heading)를 가지는데, 일반적으로 그 컬럼의 가장 위쪽에 오는 내용을 가리킨다. 표의 캡션[表題]에 적힌 내용을 볼 때 명확히 이해할 수 있는 경우라면 스터브에는 난제를 두지 않는 경우도 있다. 난제 아래 차난제(subheading)를 두는 경우가 있는데, 대개 숫자 데이터의 단위를 표시한다.

복잡한 표는 두 개 이상 수준의 난제를 가지기도 하는데, 이 때는 decked head를 사용한다. decked head는 차상난제*(spanner head)*와 거기에 딸린 둘 이상의 속난제로 이루어진다. 각 속난제들이 차상난제에 속한다는 것을 보이기 위해 난제괘선*(spanner rule)*을 차상난제와 속난제 사이에 긋는다.

이중난제, 삼중난제 등은 피해야 한다. 그렇게 하면 표의 내용이 어디에 속하는지 찾기가 어려워지기 때문이다. 이중난제를 사용하는 대신 단난제*(斷欄題, cut-in head)*를 사용할 수 있다. 단난제란 컬럼을 가로지르는 표제행으로서 그 아래 오는 내용을 모두 설명하는 난제를 가리킨다.

종선에 대해서는 아무런 언급을 하지 않았는데, 이것은 의도적인 것이다. 테이블에는 종선이 없어야 한다. 괘선을 긋는다면 횡선만을 긋는다. 그리고 괘선은 두 개 또는 세 개 이상을 겹쳐 그어서는 안된다. 오직 하나만을 긋는다. 이것은 잉크가 비싸기 때문이 아니고 실제로 조판이 수작업으로 이루어지기 때문도 아니다. 시작적인 장애요인을 제거하기 위해서이다.

일례로, 표 [12.2]의 표 [12.2(a)]는 LaTeX 교과서에서 가져온 것이다. 그리고 표 [12.2(b)]는 Simon Fear [Fea03]가 이 표에서 불필요한 선을 제거한 결과이다.

```
\begin{table}
\centering
```

표 12.2: Two views of one table

(a) Before

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

(b) After

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

```
\caption{Two views of one table} \label{tab:2views}
\subtop[Before]{\label{tab:before}%
 \begin{tabular}{||l|lr||} \hline
 gnats     & gram      & \$13.65 \\ \cline{2-3}
           & each      & .01 \\ \hline
 gnu       & stuffed   & 92.50 \\ \cline{1-1} \cline{3-3}
 emu       &           & 33.33 \\ \hline
 armadillo & frozen    & 8.99 \\ \hline
 \end{tabular}}
\hfill
\subtop[After]{\label{tab:after}%
 \begin{tabular}{@{}llr@{}} \toprule
 \multicolumn{2}{c}{Item} \\ \cmidrule(r){1-2}
 Animal & Description & Price (\$)\\ \midrule
 Gnat  & per gram  & 13.65 \\
       & each      & 0.01 \\
 Gnu   & stuffed   & 92.50 \\
 Emu   & stuffed   & 33.33 \\
 Armadillo & frozen & 8.99 \\ \bottomrule
 \end{tabular}
}
\end{table}
```

피어(Fear)의 법칙

Simon Fear는 기본 LaTeX 표 괘선에 불만을 느껴서 booktabs [Fea03] 패키지를 작성했다. 이 패키지는 더 나은 횡선을 긋도록 해준다. 다른 많은 타이포그라피 전문가들과 마찬가지로, 그도 종선은 싫어한다.

간단한 경우, 표는 \toprule로 시작한다. 여기에는 난제로 이루어진 한 행이 온다. 그런 다음에 \midrule이라는 구분선을 긋고 그 뒤에 컬럼 데이터들을 둔 다음, 마지막에 \bottomrule을 긋는다. 대부분의 조판가들은 \toprule과 \bottomrule을 가운데 구분선인 \midrule보다 굵게(즉, 두껍고 진하게) 긋는다.

```
\toprule[⟨wd⟩] \bottomrule[⟨wd⟩] \heavyrulewidth
\midrule[⟨wd⟩] \lightrulewidth
```

여기의 괘선 명령은 기본값의 두께를 취하였다. 문서에서 재설정할 수도 있다. top과 bottom 괘선에는 \heavyrulewidth가, 중간 괘선에는 \lightrulewidth가 적용되었다(아래에서 더 자세히 설명한다). 그럴 경우는 거의 없겠지만 특별한 두께가 필요하다면 ⟨*wd*⟩ 길이값 선택인자를 이용한다.

괘선 명령은 이전 행의 \\ 명령 직후에 와야 한다(예외는 \toprule이다. 이것은 \tabular{} 명령 다음에 온다). 즉, LaTeX가 \hline이나 \cline을 허용하는 위치에 와야 한다.

```
\cmidrule[⟨wd⟩](⟨trim⟩){⟨a–b⟩} \cmidrulewidth
```

일반적인 \cline 매크로와 마찬가지로, \cmidrule은 숫자로 지정하는 ⟨*a*–*b*⟩ 범위의 컬럼 사이에만 괘선을 긋는다. 일반적으로 말해서 이 괘선은 마지막 컬럼까지 완전히 채우지 않기 때문에, 특히 \cmidrule이 다른 행의 끝 직후에 와야 할 필요가 있다. 'trimming' 선택 명령은 (r), (l), (rl), (lr) 가운데 하나인데, 괘선의 오른쪽이나 왼쪽 끝을 잘라내도록 한다. 이 옵션 인자에는 중괄호나 대괄호를 쓰지 않고 괄호를 사용하고 있다는 점을 주의하라.

```
\cmidrule(r){1-2}
```

표 [12.2(b)]에는 위의 코드가 사용되었다.

기본 두께는 \cmidrulewidth이다. ⟨*wd*⟩ 선택인자를 지정하면 이것을 바꿀 수 있다.

이 명령이 사용되고 있는 보기로는 위의 표 [12.2(b)]에 나타나 있는 'after'를 보라.

이따금 마지막 행 다음에 총계를 표시하기 전과 같이 테이블의 어떤 행과 다음 행 사이에 추가적인 공백이 필요할 때가 있다. \addlinespace를 \\ 이후에 둠으로써 간단히 해결할 수 있다.

```
\addlinespace[⟨wd⟩] \defaultaddspace
```

\addlinespace는 ⟨*wd*⟩의 두께를 갖는 흰색(색깔없는) 괘선이라고 생각하는 것이 좋다. \defaultaddspace의 기본 간격은 1행간보다 적다.

`\specialrule{`⟨*wd*⟩`}{`⟨*abovespace*⟩`}{`⟨*belowspace*⟩`}`

테이블에서 이중괘선을 사용해서는 안된다. 오로지 두께가 다른 괘선만을 사용하라. `\specialrule`은 특별한 두께와 간격을 가진 괘선을 그리기 위한 것이다. 그러나 앞서 언급한 괘선만으로 충분할 것이므로 `\specialrule`을 써야 할 상황이 없기를 바란다.

`\morecmidrules`

그럼에도 불구하고, 하여튼 `\cmidrule`에 이중괘선을 치고야 말겠다면 `\morecmidrules`를 적절히 사용할 수는 있다. 두 개의 `\cmidrule` 명령을 주면 '줄이 그려지는' 행이 동일한 행이기 때문에 부적절하다.

```
\cmidrule{1-2}\cmidrule{1-2}
```

여기서 두번째 명령은 첫번째 괘선과 똑같은 위치에 그려진다.

```
\cmidrule{1-2}\morecmidrules\cmidrule{1-2}
```

이렇게 하면 1번과 2번 컬럼 사이에 두 개의 선이 그려지고, 두 선의 간격은 `\cmidrulesep`이 된다. `\morecmidrules` 명령을 주기 전에 전체 행 괘선이 그려져야 한다. `\morecmidrules` 그 자체는 이 명령 직후에 `\cmidrule`이 오지 않는다면 아무런 효과가 없다. 즉, 이것은 일반적인 간격을 설정하는 명령이 아니다.

새로운 괘선 명령은 `\hline`이나 `\cline`과 함께 사용할 수도 있다. 더 중요한 점은, 새로운 괘선 그리기 명령은 `{|}`에 의해 만들어지는 수직괘선과는 잘 연결되도록 설계되어 있지 않다. 이것은 의도적인 것으로, 수직 괘선을 테이블에서 되도록 사용하지 않도록 하기 위한 것이다.

tabular 환경

`\begin{tabular}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabular}`
`\begin{tabular*}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabular*}`
`\begin{tabularx}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabularx}`

`tabular` 환경으로 만들어진 테이블은 엔트리들이 모두 들어갈 정도의 폭으로 조판된다. 그러나 `tabular*`와 `tabularx` 환경으로 만들어진 표는 테이블의 전체 폭을 추가적인 ⟨*width*⟩ 인자로 설정해줄 수 있다.

`tabular*` 환경은 컬럼 사이의 공간을 변경함으로써 필요한 길이를 조절한다. 반면 `tabularx` 환경은 컬럼 폭을 변경한다. 조정이 필요한 컬럼은 `X` 컬럼 지시자로 ⟨*preamble*⟩에 지정하는데, `X` 컬럼은 폭 계산이 완료되면 `p` 컬럼으로 바뀐다.

다음 코드는 정상적인 tabular이다.

```
\begin{center}
\begin{tabular}{|c|p{5.5pc}|c|p{5.5pc}|}  \hline
\multicolumn{2}{|c|}{Multicolumn entry!} & THREE & FOUR \\ \hline
one &
\raggedright\arraybackslash The width of this column is fixed (5.5pc). &
three &
\raggedright\arraybackslash Column four will act in the same way as
  column two, with the same width.\\
\hline
\end{tabular}
\end{center}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.

다음 보기는 내용이 동일하지만 각 환경의 설정 방법을 바꾸어 보았다.
먼저 tabularx를 이용하여 다음과 같이 설정하여 보았다.

```
\begin{tabularx}{250pt}{|c|X|c|X|}
```

결과는 다음과 같다.

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.[1]	three	Column four will act in the same way as column two, with the same width.

다음은 tabular*를 이용한 예이다.

[1] You can use footnotes in a tabularx!

```
\begin{tabular*}{250pt}{|c|p{5.5pc}|c|p{5.5pc}|}
```

여기서는 X 컬럼이 없다는 점에 주의하라. tabular의 전체 폭이 지정된 폭보다 적기 때문에 횡선이 오른쪽으로 넘어가서 tabular의 지시된 폭 만큼 그어졌다.

Multicolumn entry!		THREE	FOUR	
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.	

다음 tabular* 테이블 정의를 보자.

```
\begin{tabular*}{300pt}{|@{\extracolsep{\fill}}c|p{5.5pc}|c|p{5.5pc}|}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column's text is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.

아래 tabularx는 다음과 같이 지시되었다.

```
\begin{tabularx}{300pt}{|c|X|c|X|}
```

결과는 다음과 같다.

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.

tabularx와 tabular* 환경의 주요한 차이점은 다음과 같다.

- tabularx는 컬럼의 폭을 조절하는데 반해, tabular*는 컬럼 사이의 공백을 조절한다.
- tabular와 tabular* 환경은 한 환경 안에 새로운 환경을 넣어도 된다. 그러나 만약 tabularx 환경 하나가 다른 tabularx 안에 들어가게 된다면 안으로 들어간 환경은 반드시 { }로 둘러싸주어야 한다.
- tabularx 환경의 내용은 실은 한 명령에 주어지는 인자이다. 따라서 명령의 인자에 허용되지 않는 구문(\verb 같은 것)은 사용할 수 없다.[2]
- tabular*는 TeX의 primitive를 사용하여 컬럼 사이의 공백을 수정한다. tabularx는 적절한 컬럼 폭이 될 때까지 테이블을 반복해서 설정한다. 그러므로 tabularx가 훨씬 더 느리다. 또한 테이블 내용이 TeX 지시어에 일치할 때까지 여러 번에 걸쳐 확장한다.

\tracingtabularx

\tracingtabularx 선언 이후에는 그 뒤에 오는 모든 tabularx 환경에 대하여, 적절한 폭을 찾을 때까지 테이블을 반복해서 재설정한 기록을 남긴다.

X 지시자는 스스로 p{⟨*some value*⟩}로 전환된다. 컬럼 폭이 좁은 경우에는 특별한 포맷을 요구하는데, 이것은 > 구문을 이용하여 설정할 수 있다. 예컨대, >{\small}X와 같이 할 수 있는 것이다. 폭이 좁은 컬럼에서는 ragged right가 필요할 수도 있다. 그러나 LaTeX의 \raggedright 매크로는 \\를 재정의하므로, tabular나 array 환경에서 충돌이 일어난다.

\arraybackslash

이런 이유 때문에 \arraybackslash 명령이 제공된다. 이것은 \raggedright나 \raggedleft, \centering 선언 뒤에 사용할 수 있다. 따라서 tabularx 설정부는 다음과 같은 내용을 포함하게 된다.

```
>{\raggedright\arraybackslash}X
```

이 설정부 설정은 \newcolumntype으로 정의하여 쓸 수 있다. 이를테면 다음과 같이 정의한 다음에,

```
\newcolumntype{Y}{>{\small\raggedright\arraybackslash}X}
```

[2]사실, verb와 verb*를 사용하면 공백을 부적절하게 취급하게 된다. 그리고 짝이 맞지 않는 {이나 }, 또는 % 문자를 사용할 수 없다.

Y를 tabularx 설정부 인자로 사용하면 될 것이다.

\tabularxcolumn

X 컬럼은 p 컬럼을 사용하도록 설정되어 있다. 이것은 \parbox[t]에 해당한다. m 컬럼을 이용하여 \parboxc에 해당하는 박스를 만들고 싶을 때가 있을 것이다. 여기서는 > 구문을 이용하여 컬럼 타입을 바꿀 수 없으므로, 다른 방법이 제공된다. \tabularxcolumn 명령은 한 개의 인자를 가진 정의할 수 있는 명령인데, 이것은 확장되어 X에 상당하는 설정부 설정으로 tabular에서 쓰이게 된다. 주어진 인자는 컬럼 폭을 계산하도록 치환된다.

기본 정의는 다음과 같이 되어 있는데,

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

이것을 다음과 같이 바꾸어보자.

```
\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}
```

그러면 X 컬럼이 m 컬럼 형식에 \small 폰트를 사용하도록 고쳐질 것이다.

모든 X 컬럼은 모두 같은 폭을 갖도록 되어 있다. 그러나 필요하다면 tabularx가 폭을 계산할 때 다른 값을 갖게 만들도록 할 수도 있다. 설정부에 다음과 같이 써보자.

```
{>{\hsize=.5\hsize}X>{\hsize=1.5\hsize}X}
```

이것은 두 개의 컬럼을, 두번째 컬럼이 첫번째 컬럼의 세 배 폭을 갖도록 만든 경우이다. 이런 일을 꼭 하고 싶다면 다음에 제시하는 규칙을 따라야 한다.

1. 모든 X 컬럼의 폭의 합이 바뀌지 않도록 해야 한다. 위의 예에서는 컬럼이 두 개였기 때문에 폭의 합이 2가 되도록 하였다.
2. \multicolumn 엔트리를 X 컬럼에 걸쳐서 지정하지 않아야 한다.

다른 모든 규칙이 그러하듯이, 규칙은 깨지기 위해 있는 것이다.

테이블의 '정상' 컬럼의 폭이 이미 필요한 폭의 합보다 커져 있을 수도 있다. tabularx는 X 컬럼이 음수값을 갖도록 허용하지 않는다. 그러므로 이럴 경우에 "X Columns too narrow (table too wide)"라는 경고를 만나게 된다.

이럴 경우 X 컬럼은 1em 폭으로 설정된다. 따라서 테이블 자체의 폭은 그 환경의 인자에서 지정된 폭의 합계보다 더 넓어지게 된다.

표준 \verb 매크로는 tabularx 안에서 작동하지 않는다. 이것은 이 명령이 어떤 매크로의 인자로도 작동하지 않는 데서 온 것이다.

\TX@verb

\TX@verb는 '불쌍한 사람을 위한 \verb'로서, TeXBook 382페이지에 기초한 것이다. 거기에서 설명하고 있듯이, 이런 방식으로 다루어지는 verbatim은 스페이스가 적절하게 처리되지 않는다. 그러므로 \verb*는 의미가 없다. 이 방법은 일반적으로 사용할 수 있는 것으로서 특정의 매크로에 \verb 형식이 포함된 인자를 허용하고 싶다면 다음과 같이 지시한다.

```
\let\verb=\TX@verb
```

나중에 원래 정의를 회복해야 한다는 것을 잊어서는 안된다.

\verb를 이런 식으로 사용할 때 적용되는 제한은 다음과 같다.

1. 인자 속의 스페이스는 verbatim으로 처리되지 않는다. 따라서 TeX의 일반적인 처리방법에 따라 skip된다.
2. 입력시에 스페이스를 주지 않았다 하더라도 제어 명령 이후에는 스페이스가 추가되어 나타난다.
3. 인자 속에 스페이스가 하나가 아니고 그 뒤에 이어지는 스페이스가 있다면, 원래 입력이 어떠했든 간에 (2)에서 말한 바에 따라 처리되지 않는다.
4. 인자는 \ 문자로 끝나면 안된다. 따라서 \verb|\|는 허용되지 않는다. 그렇지만 (3)에 의하여 \verb|\ |는 \으로 식자된다.
5. {와 }가 짝을 맞추어야 한다. 그러므로 \verb|{|는 허용되지 않는다.
6. 주석 문자 %는 verbatim으로 나타나지 않는다. 보통 주석 문자와 같이 그 이후의 행을 주석처리할 것이다.
7. ?‘, !‘과 같은 결합은 cmtt 폰트를 사용하는 경우 ¿, ¡와 같이 나타날 것이다.

12.6 부정형 테이블

모든 tabular 환경은 테이블을 하나의 박스에 넣는다. 즉, LaTeX은 이 표 전체를 하나의 커다랗고 복잡한 글자처럼 간주한다. 글자는 여러 페이지에 나누어서 식자될 수 없다. 페이지 바닥을 넘치는 긴 표를 그리려 한다면, longtable [Car98b]이나 xtab [Wil00e]과 같은 패키지를 사용해야 한다. 이 패키지들은 페이지 경계에 이르면 자동으로 테이블을 쪼갠다. 여기에는 여러 가지 부수적인 작업이 필요하다. 즉, 캡션을 각 페이지 상단에 붙여야 하고 난제를 다시 적어주어야 하는 등의 일이 필요한 것이다.

이어지는 tabular

\begin{ctabular}[⟨*pos*⟩]{⟨*preamble*⟩} rows \end{ctabular}
\begin{ctabular*}[⟨*pos*⟩]{⟨*width*⟩}{⟨*preamble*⟩} rows \end{ctabular*}

ctabular 환경은 tabular와 비슷하다. 차이점도 좀 있는데, 중요한 것은 여러 페이지에 걸쳐서 식자될 수 있다는 점이다. ⟨*preamble*⟩ 인자는 앞서 설명한 array와 tabular 환경의 경우와 동일하다. 그러나 ⟨*pos*⟩ 옵션 인자는 페이블의 수직 위치가 아닌 수평 위치를 제어한다. 여기에 올 수 있는 인자값은 다음과 같다. l (왼쪽 정렬), c (중앙 정렬), r (오른쪽 정렬). 기본값은 c이다.

```
\begin{ctabular}{lcr}  \toprule
LEFT & CENTER & RIGHT \\  \midrule
l & c & r \\
l & c & r \\
l & c & r \\
l & c & r \\  \bottomrule
\end{ctabular}
```

LEFT	CENTER	RIGHT
l	c	r
l	c	r
l	c	r
l	c	r

ctabular 환경은 table 환경 안에 올 수 없다. 이렇게 하면 테이블이 다음 페이지로 잘라지지 못하게 된다. ctabular의 캡션은 fixed caption으로 정의할 수 있다.

```
\newfixedcaption{\freetabcaption}{table}
```

위와 같이 정의한 다음, table 플로트 안에서 \caption을 쓰듯이 \freetabcaption을 사용하면 된다.

12.7 자동 tabular

tabular 형식은, 예컨대 특정 조직 구성원의 이름이나 LaTeX 환경의 명칭을 나열하는 데 쓰일 수 있다.

문서의 초안을 작성할 때, 또는 엔트리가 바뀌기 쉬울 때는 특히 나열할 항목들을 행끝 문자를 일일이 명시하지 않고도 표 형식으로 배열할 수 있다면 좋을 것이다.

\autorows[⟨*width*⟩]{⟨*pos*⟩}{⟨*num*⟩}{⟨*style*⟩}{⟨*entries*⟩}

\autorows 매크로는 ⟨*entries*⟩를 행으로 나열하는 매크로이다. 즉, 엔트리들을 왼쪽에서 오른쪽으로, 위에서 아래로 차례로 조판한다. 다음은 \autorows를 이용하여 만든 표이다.

```
\freetabcaption{Example automatic row ordered table}
\autorows{c}{5}{c}{one, two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
```

표 12.3: Example automatic row ordered table

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

⟨*pos*⟩ (l, c, r) 인자는 표의 수평 위치를 제어하고, ⟨*num*⟩은 컬럼의 수이다. ⟨*style*⟩ (l, c, r) 인자는 엔트리가 컬럼에 놓일 방식을 지시한다. 각 컬럼은 모두 동일하게 처리된다. ⟨*entries*⟩ 인자에는 쉼표로 구분된 명칭들을 넣는데, 이것이 표로 처리된다. 마지막 항목과 닫는 중괄호 사이에는 쉼표가 오면 안된다.

각 컬럼 폭은 열거항목 중에서 제일 긴 것을 넣을 수 있을 정도로 하되 동일한 값을 가진다. ⟨*width*⟩ 값이 양수(예, [0.8\textwidth])이면, 전체 표의 폭을 지시하는데, 각 컬럼 폭은 이 값을 컬럼 수로 나누어서 구한다. 음수이면 ⟨*width*⟩ 값이 그 컬럼에서 제일 긴 항목이 들어갈 수 있는 값으로 각 컬럼 폭을 설정하도록 하는 것이다. 여기서는 컬럼 폭이 정해진 값이 아니다.

아래 예제는 ⟨*width*⟩ 인자에 따라 어떤 효과가 나타나는지를 보인 것이다. 기본값은 0pt이다.

```
\freetabcaption{Changing the width of a row ordered table}
\autorows[-1pt]{c}{5}{c}{one, two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
\autorows[0pt]{c}{5}{c}{one, two, three, four, five, ... }
\autorows[0.9\textwidth]{c}{5}{c}{one, two, three, four, five, ... }
```

표 12.4: Changing the width of a row ordered table

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

one	two	three	four	five
six	seven	eight	nine	ten

	eleven	twelve	thirteen	fourteen
one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

`\autocols[`⟨*width*⟩`]{`⟨*pos*⟩`}{`⟨*num*⟩`}{`⟨*style*⟩`}{`⟨*entries*⟩`}`

`\autocols` 매크로는 ⟨*entries*⟩를 세로로 배열하는데, 순서는 위에서 아래로, 왼쪽으로 오른쪽으로이다. ⟨*width*⟩를 제외하고 인자는 `\autorows`의 경우와 같다. 컬럼 폭은 전체 테이블에 대하여 언제나 고정된 값이며 가장 긴 엔트리가 들어가기에 충분한 값으로 정해진다. ⟨*width*⟩가 양수이면 `\autorows`의 경우와 동일한 결과를 가져오지만, 음수이면 무시된다.

`\autorows`나 `\autocols`를 사용할 때 엔트리에 쉼표를 넣어야 할 때가 있을 것이다. 다음과 같이 한다.

```
\newcommand*{\comma}{,}
```

다음 예제를 보라.

```
\freetabcaption{Changing the width of a column ordered table}
\autocols{c}{5}{c}{one\comma{} two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
\autocols[0.9\textwidth]{c}{5}{c}{one\comma{} two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
```

표 12.5: Changing the width of a column ordered table

one, two	five	eight	eleven	thirteen
three	six	nine	twelve	fourteen
four	seven	ten		

one, two	five	eight	eleven	thirteen
three	six	nine	twelve	fourteen
four	seven	ten		

12.8 간격조절

이따금 tabular의 행이 너무 바짝 붙는 경우가 있다.

\extrarowheight

\extrarowheight라는 이름의 길이변수는 tabular와 array의 행 높이를 수정하도록 해준다. 이 값이 양수이면 그 값만큼 각 행의 높이(height)에 더해지고 깊이(depth)는 그대로 유지된다. 이것은 특히 가로선이 있는 표에서 중요한데, 이 선이 일반적으로 대문자에 걸리기 때문이다.

\hline의 특별한 변형

tabular류의 환경들은 그 환경이 표현되는 텍스트의 베이스라인을 기준으로 수직 위치가 결정된다. 기본값은 (세로) 중앙정렬이지만, t나 b 값을 옵션 위치 인자로 지정함으로써 첫 행을 베이스라인에 두도록 하거나 마지막 행을 베이스라인에 두도록 바꿀 수 있다. 그러나, 이것이 환경의 첫번째 요소가 \hline 명령일 때는 작동하지 않는다. 이 경우에는 괘선을 베이스라인에 일치시켜서 정렬된다.

다음 예를 보자.

Tables with no hline commands used versus tables with some hline commands used.

```
Tables
\begin{tabular}[t]{l}
 with no\\ hline \\ commands \\ used
\end{tabular} versus tables
\begin{tabular}[t]{|l|}
 \hline
  with some \\ hline \\ commands \\
 \hline
\end{tabular} used.
```

\firsthline \lasthline \extratabsurround

\firsthline과 \lasthline을 이용하면 이 문제를 해결할 수 있다. 테이블이 엄청나게 많은 내용을 표함하고 있지 않다면 첫째줄과 마지막줄이 적절하게 배열된다.

Tables with no line commands used versus

```
Tables
\begin{tabular}[t]{l}
  with no\\ line \\ commands \\ used
\end{tabular} versus tables
\begin{tabular}[t]{|l|}
 \firsthline
  with some \\ line   \\ commands \\
 \lasthline
\end{tabular} used.
```

tables | with some line commands | used.

이 두 명령을 구현하기 위해서 추가적인 길이변수를 사용하였다. `\extratabsurround`라는 이 길이변수는 환경의 처음과 끝에 약간의 여분 공간을 추가한다. 이렇게 하는 것이 특히 테이블 안에 다시 tabular가 올 때 유용하다.

괘선

테이블 안의 수평 수직 괘선을 긋는 데는 두 가지 방법이 있을 수 있다.

1. 테이블의 크기를 변화시키지 않고 가능한 위치에 괘선을 두는 방법
2. 컬럼과 행 사이에 오는 괘선 자체가 폭을 갖게 함으로써 결과적으로 테이블을 늘리는 방법

이 클래스는 두번째 방법을 채택하였다. 이것은 LaTeX 커널이 구현하고 있는 첫번째 접근방법과는 다른 것이다.

표준 LaTeX에서는 테이블에 괘선을 추가해도 테이블 자체의 폭과 길이는 변하지 않는다(다만 이중괘선의 경우는 예외이다). 즉, 설정부에서 `lll`로 한 경우나 `l|l|l`로 한 경우나 문서 모양에는 아무런 변화를 주지 않는다. 다만 괘선이 추가될 따름이다. 반면, 이 클래스에서 테이블이 `\textwidth`에 맞추어져 있다면 괘선을 추가하면 overfull 박스를 만들게 된다. (아무튼지, 수직선을 테이블에서 사용해서는 안된다.)

13 여백과 각주

표준 클래스는 \marginpar 명령을 제공한다. 이 명령은 여백에 어떤 내용을 둔다. 이 클래스는 두 개의 추가적인 여백 주석을 지원한다. 그리고 각주 기능을 좀더 확장하고 있다.

13.1 각주(footnote)

각주는 페이지 바닥에 오는 특별한 종류의 플로트라고 볼 수 있다.

\footnote[⟨*num*⟩]{⟨*text*⟩} \footnotemark[⟨*num*⟩] \footnotetext[⟨*num*⟩]{⟨*text*⟩}

%본문에서 \footnote 명령은 그 위치에 각주표지를 두고 ⟨*text*⟩의 앞머리에 같은 표지를 붙여서 페이지의 바닥에 놓는다. ⟨*num*⟩ 옵션 인자가 주어지면 그 값이 표지에 사용된다. 그렇지 않으면 footnote 카운터를 증가시켜서 표지의 값으로 사용한다.

\footnotemark 명령으로 본문에 표지만을 둘 수 있다. 이 값은 \footnote의 경우와 동일하게 결정된다. 각주 텍스트는 \footnotetext 명령으로 바닥에 놓인다. ⟨*num*⟩ 옵션 인자가 주어지면 표지의 번호로 사용하고 그렇지 않으면 footnote 카운터 값을 사용한다. 이를테면 \footnote 명령이 다음과 같이 정의된 것으로 생각하면 이해가 쉬울 것이다. 완전하게 동일하지는 않지만.

```
\newcommand{\footnote}[1]{\footnotemark\footnotetext{#1}}
```

\footref{⟨*label*⟩}

이따금 각주 텍스트에 한 개 이상의 참조를 매기고 싶은 경우가 있다. 이 때는 \label을 각주에 두고 \footref 명령으로 그 label을 참조하면 각주 label을 찍어준다. 예를 들면 다음과 같다.

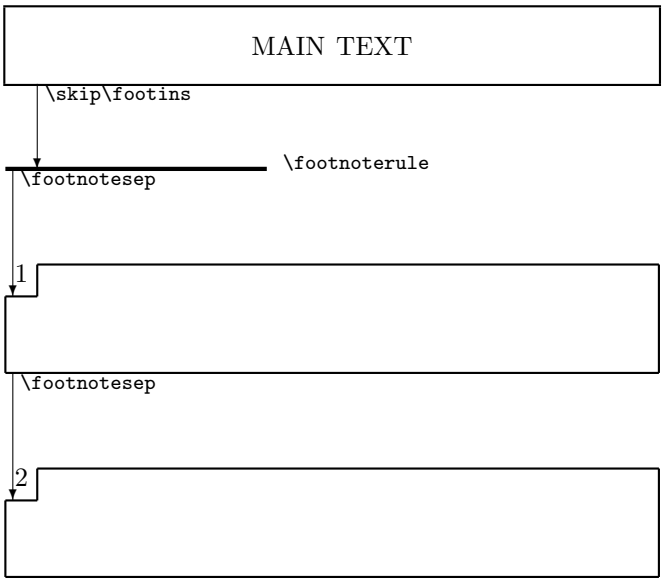


그림 13.1 — 표준 클래스의 각주 레이아웃 파라미터

```
...\footnote{... adults or babies.\label{fn:rabbits}}
...
... The footnote\footref{fn:rabbits} on \pref{fn:rabbits} ...
```

위의 마지막 행은 다음과 같이 나타난다.

... The footnote[3] on 15 쪽 ...

```
\footnotesep
\skip\footins
```

`\footnotesep` 길이는 각주(및 thanks 주석) 사이의 거리를 결정한다. 이 값은 `\footnotesize` 폰트에 추가적인 공백을 주지 않도록 초기화되어 있는데, 값을 바꾸려면 다음처럼 한다.

```
\addtolength{\footnotesep}{...}
```

`\skip\footins` 길이는 본문의 바닥에서 첫번째 각주의 첫 줄 까지 거리를 지정한다. 마찬가지로 다음과 같이 바꾼다.

```
\addtolength{\skip\footins}{...}
```

본문과 각주 첫째줄의 베이스라인 사이의 길이의 합은 \footnotesep + \skip\footins 이다.

\footnoterule

\footnoterule 매크로는 LaTeX 커널에 정의되어 있는 것으로 표준 클래스에서 재정의하고 있다. '@' 문자 없이 클래스 정의를 조금 변형하면 다음과 같이 되는데,

```
\renewcommand{\footnoterule}{%
   \kern -3pt                    % call this kerna
   \hrule height 0.4pt width 0.4\columnwidth
   \kern 2.6pt                   % call this kernb
}
```

이것은 왼쪽 마진에서 \columnwidth의 40% 길이로 0.4pt 두께의 선을 긋는 것이다. 이 선은 수직 간격을 차지하지 않으므로 kerna + kernb는 선의 두께와 같아야 한다. 이 선은 본문의 바닥에서 \skip\footins~+~karna 길이만큼 떨어져서 위치한다. 그러므로 이 선을 위로 이동시키려면 kerna 값을 줄이고 kernb 값을 증가시킨다. 아래로 이동시키려면 그 반대로 하면 된다.

이 클래스에서 상당한 추가 파라미터를 제공하고 있지만, 좀더 다양하게 각주 스타일을 구현하고 싶으면 footmisc 패키지 [Fai00]를 사용할 수 있다. 이 클래스의 추가 기능은 footmisc 패키지와 충돌하지 않도록 정의되었다. 87 쪽의 §7.2에 설명된 \thanks에도 비슷한 발상이 적용되었다.

\footnote 매크로는 결과적으로 세 가지 일을 한다.

- \footnote가 불린 위치에 각주표지를 표시하는 것.
- \footnote가 불린 페이지의 바닥에 각주표지를 표시하는 것.
- 페이지 바닥의 각주표지 뒤에 각주 텍스트를 식자하는 것.

\@makefnmark

\footnote 매크로는 \@makefnmark라는 커널 명령을 불러서 각주표지를 \footnote가 불린 위치에 식자한다(표지의 값은 \@thefnmark 매크로에 저장되어 있다). 기본값은 표지를 상첨자로 식자하는 것이다.

```
\def\@makefnmark{\hbox{\textsuperscript{\@thefnmark}}}
```

만약 이 표지를 보통 글꼴로 찍고 각진괄호로 둘러싸고자 한다면,

```
\renewcommand*{\@makefnmark}{ [\@thefnmark]}
```

이와 같이 정의한다.

\footfootmark
\footmarkwidth \footmarkstyle{⟨*arg*⟩}

이 클래스에서 페이지 바닥에 각주표지를 찍는 매크로는 \footfootmark이다. 이 표지는 \footmarkwidth 폭을 갖는 박스로 식자된다. 이 값이 음수이면 표지는 마진쪽으로 빠져나간다. 영(0)이면 flush left되고, 양수이면 들여쓰기된다. 표지의 모양은 \footmarkstyle에 의하여 결정하는데, 기본값은 다음과 같다.

```
\footmarkstyle{\textsuperscript{#1}}
```

여기서 #1은 \@thefnmark이 놓일 자리이다. 이 기본 정의는 상첨자로 식자하게 한다. 예를 들어 표지를 베이스라인에 놓고 오른쪽 괄호로만 표시하고 싶다면,

```
\footmarkstyle{#1) }
```

이렇게 정의한다.

\footmarksep \footparindent

표지는 \footmarkwidth 폭을 가진 박스로 식자된다. 그런 다음에 각주 텍스트가 온다. 각주 텍스트의 두번째 이후의 행은 박스의 끝에서 \footmarksep 만큼 떨어진 위치에 놓인다. 각주 문단의 첫번째 줄은 \footparindent 만큼 들여쓴다.

\foottextfont

각주 텍스트는 \foottextfont 폰트로 식자한다. 기본값은 \footnotesize이다.

종합하면, 이 클래스의 정의는 다음과 같다.

```
\footmarkstyle{\textsuperscript{#1}}
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-1.8em}
\setlength{\footparindent}{1em}
\newcommand{\foottextfont}{\footnotesize}
```

이것은 표준적인 각주 레이아웃을 구현한 것이다.

다음과 같은 \footmarkwidth와 \footmarksep 조합으로 자신이 원하는 모양을 만들어볼 수 있을 것이다. 첫번째 조합이 표준 레이아웃이다.

```
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{-1.8em}
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{0em}    \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{-1.8em} \setlength{\footmarksep}{1.8em}
\setlength{\footmarkwidth}{0em} \setlength{\footmarksep}{1.8em} \footmarkstyle{#1)\hfill}
```

\makefootmarkhook

각주를 만들기 직전에 \makefootmarkhook 매크로가 불린다. 기본값은 아무런 일도 하지 않지만 재정의하여 쓸 수 있다. 예를 들면 다음과 같다.

```
\renewcommand{\makefootmarkhook}{\raggedright}
```

위의 정의는 각주가 raggedright로 만들어지게 할 것이다.

이 이후의 각주에 대해서는 다음과 같은 설정을 적용하겠다.

```
\setlength{\footmarkwidth}{-1.0em}
\setlength{\footmarksep}{-\footmarkwidth}
\footmarkstyle{#1}
```

\multfootsep

표준 클래스에서는 둘 또는 그 이상의 각주를 연속해서 붙이고자 할 때,[1,2] 본문의 각주 표지가 합쳐져서 식자된다. 그러나 이 클래스에서는 footmisc [Fai00]와 ledmac [Wil03] 패키지에서와 같이 두 표지 사이에 구분자를 삽입한다. 이 클래스에서는 \multfootsep이 구분자로 쓰이는데, 기본 정의는 다음과 같다.

```
\newcommand*{\multfootsep}{\textsuperscript{\normalfont,}}
```

1 각주 한 개
2 또 한 개의 각주가 이어짐

```
\feetabovefloat
\feetbelowfloat
```

표준 클래스에서 각주가 붙는 페이지에 바닥 플로트가 오면, 각주가 플로트 위쪽에 식자된다. 이것은 이상해 보인다고 생각한다. \feetbelowfloat 선언을 하면 각주가 bottom 플로트보다 아래쪽 페이지 바닥에 식자되도록 할 수 있다. 또 \raggedbottom의 경우 각주를 텍스트의 마지막 행 직후에 식자하지 않고 bottom까지 떨어뜨려서 둔다. 표준 위치는 \feetabovefloat 선언을 따르는 것이고 그것이 기본값이다.

```
\plainfootnotes
\twocolumnfootnotes
\threecolumnfootnotes
\paragraphfootnotes
```

일반적으로, 각각의 각주는 새로운 문단이 된다. 이 클래스는 세 가지 종류의 스타일을 더 제공하므로 모두 네 종류의 형식이 있다. \twocolumnfootnotes 선언은 각주를 2단으로 조판한다. 마찬가지로 \threecolumnfootnotes 선언은 3단 조판을 한다. \paragraphfootnotes 선언 이후에는 여러 각주들이 하나의 문단을 이루도록 식자된다. 기본 형식은 \plainfootnotes 선언이다.

이 스타일은 언제라도 바꿀 수 있다. 그러나 그 변경이 한 페이지의 중간에서 이루어지고 선언의 전후에 각주가 존재한다면 신기한 결과를 얻을 수도 있다. \maketitle과 \thanks, 그리고 타이틀이 있는 페이지에 각주도 존재하는 article 형식의 문서에서 스타일을 변경해보는 것도 재미있다.

```
\title{...\thanks{...}}
\author{...\thanks{...}...}
...
\begin{document}
\paragraphfootnotes
\maketitle
\plainfootnotes
...
```

```
\newfootnoteseries{⟨series⟩}
\plainfootstyle{⟨series⟩}
\twocolumnfootstyle{⟨series⟩}
\threecolumnfootstyle{⟨series⟩}
\paragraphfootstyle{⟨series⟩}
```

이 클래스는 서로 다른 계열을 갖는 각주도 정의하여 사용할 수 있다. 일반적인 각주가 있고, 아라비아 숫자로 표시한 각주와 로만 숫자로 표시한 각주가 별도로 필요하다고 하자. 새로운 계열의 각주는 \newfootseries 매크로로 만든다. 여기서 ⟨*series*⟩는 이 시리즈에 대한 식별자를 알파벳 문자로 붙인 것이다. 알파벳 대문자 하나만을 사용해서, 예를 들면 P로 하는 것이 편리하다.

여기서 \newfootnoteseries{Q} 하면, 일련의 \footnote에 해당하지만 ⟨*series*⟩ 명칭을 끝에 붙인 매크로들을 만들어준다. 즉, \footnoteQ, \footnotemarkQ, \footnotetextQ 등을 사용할 수 있게 되는 것이다. 이 명령을 \footnote와 마찬가지로 사용하면 된다.

새로 정의된 계열의 각주는 일반적인 각주별 문단 형식으로 식자된다. 특정 계열의 각주의 스타일을 바꾸려면 \...footstyle 명령을 사용한다.

예를 들어, 'P' (paragraph) 시리즈의 각주에 대해, 로만 숫자를 각주표지로 사용하면서, 본문에는 닫힌 괄호가 붙는 상첨자로 표시하고, 각주 자체에는 baseline에 내려놓으면서 en-dash를 붙이면서, 각주 텍스트는 normal footnote size의 이탤릭 글꼴로 식자하게 하려면,

```
\newfootnoteseries{P}
\paragraphfootstyle{P}
\renewcommand{\thefootnoteP}{\roman{footnoteP}}
\footmarkstyleP{#1--}
\renewcommand{\@makefnmarkS}{\hbox{\textsuperscript{\@thefnmarkP)}}}
\renewcommand{\foottextfont}{\itshape\footnotesize}
```

위와 같이 정의한 다음, 다음과 같이 사용할 수 있다.

```
.... this sentence\footnoteP{A 'p' footnote\label{fnp}}
includes footnote~\footrefP{fnp}.
```

미니페이지 각주는 별도의 카운터(mpfootn...)를 가진다. 그리고 알파벳으로 각주 번호를 표시하는 것이 기본값이다. 이것을 'P' 시리즈의 숫자로 바꾸려면, 다음과 같이 정의한다.

```
\renewcommand{\thempfootnoteP}{\arabic{mpfootnoteP}}
```

\newfootnoteseries 매크로는 각주 관련 길이 명령, 예를 들면 \footmarkwidth, \footmarksep 들을 계열별로 따로 만들지 않는다. 그리고 \footnoterule도 별도로 만들지 않는다.

페이지 하단에 일반적인 각주가 식자된 다음에, 각 그룹별로, 첫번째 계열의 각주군, 두번째 계열의 각주군 등의 순서로 모아져서 식자된다. 그룹의 순서는 \newfootnoteseries 명령이 사용된 순서를 따른다.

`\footfudgefiddle`

문단형 각주에서 TeX은 그 문단이 차지할 공간을 어림잡아야 한다. 불행히도 이것은 어림짐작일 뿐이므로, 만약 이 값을 낮추어 잡으면 각주가 페이지의 너무 아래쪽으로 가버릴 위험이 있다. 즉, 페이지 번호보다 아래쪽에 각주가 놓일 수 있는 것이다. `\footfudgefiddle` 값을 증가시키면 TeX에게 더 많은 공간을 확보하라고 할 수 있는데, 기본값은 64이므로 예컨대 다음과 같이 할 수 있다.

```
\renewcommand{\footfufgefiddle}{70}
```

이런 식으로 하는 것이 문제를 해결하는 한 가지 방법이 될 수 있다. 직접 해보면서 더 좋은 값을 찾아보아야 할 것이다.

`\fnsymbol{`⟨*counter*⟩`}` `\@fnsymbol{`⟨*num*⟩`}`

`\fnsymbol` 매크로는 해당 카운터 ⟨*counter*⟩를 각주 기호문자로 식자한다. 내부적으로 `\@fnsymbol` 매크로를 사용하는데, 이것은 양의 정수 ⟨*num*⟩을 기호문자로 바꾸는 역할을 하는 것이다. 만약 각주 기호문자의 표준적인 순서가 마음에 들지 않는다면, 이 매크로를 바꾸어야 한다. 원래 정의는 다음과 같이 되어 있다.

```
\def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or
  \mathsection\or \mathparagraph\or \|\or **\or \dagger\dagger
  \or \ddagger\ddagger \else\@ctrerr\fi}}
```

이것은 `\@fnsymbol{1},...\@fnsymbol{9}`까지 다음 기호들을 식자한다. fnsymbol1, fnsymbol2, fnsymbol3, fnsymbol4, fnsymbol5, fnsymbol6, fnsymbol7, fnsymbol8, and fnsymbol9.

Robert Bringhurst는 다음이 전통적인 순서(최소한 ¶까지는)라고 하고 있다. ∗, †, ‡, §, ‖, ¶, ∗∗, ††, ‡‡. 이 순서대로 기호를 얻으려면 다음과 같이 `\@fnsymbol`을 재정의하라.

```
\renewcommand{\@fnsymbol}[1]{\ensuremath{%
  \ifcase#1\or *\or \dagger\or \ddagger\or
  \mathsection\or \|\or \mathparagraph\or **\or \dagger\dagger
  \or \ddagger\ddagger \else\@ctrerr\fi}}
```

이 코드를 preamble에 두려면 `\makeatletter`, `\makeatother`로 둘러싸야 한다는 점을 잊어서는 안된다. 다른 저자들이나 출판사에서는 이와는 다른 순서를 선호할 수도 있다.

13.2 verbatim 각주

\verbfootnote 매크로는 verbatim 텍스트가 포함된 각주를 식자할 수 있다.

\verbfootnote[⟨*num*⟩]{⟨*text*⟩}

아래 두 단락은 다음 코드로 식자한 것이다.

```
아래, 각주\footref{fn1}\는 \cmd{\footnote}이고, 각주~\ref{fn2}\는
\cmd{\verbfootnote}이다.

\cmd{\verbfootnote} 명령은 \cmd{\footnote}로 한 것과 동일한
결과\footnote{만약 color가 사용되면 몇 가지 문제를 불러올 수
  있다.\label{fn1}}를 보이지만, verbatim
텍스트\verbfootnote{\verb?\footnote? 매크로가 \verb?\verbfootnote?와
  다른 점은 인자에 verbatim 텍스트가 포함시킬 수 없다는
  점이다.\label{fn2}}를 \meta{text} 인자에 넣을 수 있다는 점이
특징이다.
```

아래, 각주[3]은 \footnote이고, 각주 4는 \verbfootnote이다.

\verbfootnote 명령은 \footnote로 한 것과 동일한 결과[3]를 보이지만, verbatim 텍스트[4]를 ⟨*text*⟩ 인자에 넣을 수 있다는 점이 특징이다.

13.3 사이드바

\sidebar는 시험용으로 만든 명령이라고 이미 언급했지만, 여러 경우에 성공적으로 사용될 수 있다.

\sidebar{⟨*text*⟩}
\sidebarfont

\sidebar 명령은 \marginpar 명령과 비슷하다. ⟨*text*⟩ 인자를 \sidebarfont로 지정된 폰트로 마진에 식자한다. \marginpar와 다른 점은, ⟨*text*⟩가 페이지의 상단 근처에서 시작하고 만약 한 페이지에 넣기에는 너무 길다면 다음 페이지까지 이어진다는 것이다.

\sidebarfont의 기본 정의는 다음과 같다.

```
\newcommand{\sidebarfont}{\normalfont}
```

3 만약 color가 사용되면 몇 가지 문제를 불러올 수 있다.

4 \footnote 매크로가 \verbfootnote와 다른 점은 인자에 verbatim 텍스트가 포함시킬 수 없다는 점이다.

\sidebarform

사이드바는 좁은 경우가 대부분이라 raggedright로 텍스트를 조판한다. 더 정확하게 말하면, 텍스트는 다음과 같이 정의된 \sidebarform에 따라 식자된다.

```
\newcommand{\sidebarform}{\rightskip=\z@ \@plus 2em}
```

이것은 raggedright로 설정되기는 했지만 raggedness 허용치가 \raggedright보다 적다. 일반적으로 \sidebarform은 다음 예와 같이 변경할 수 있다.

```
\renewcommand{\sidebarform}{}           % 양쪽 정렬
```

만약 \sidebar 명령이 페이지 나눔 부근에서 주어지거나, 사이드바 텍스트가 본문의 비정규적인 행나눔일 일어나는 부분에 걸쳐지면(예, \section 전후), 문제가 발생할 수 있다. 또한, 문서의 끝 부분에서는 사이드바의 내용이 식자되지 않는 경우도 있다.

\sidebarwidth
\sidebarhsep
\sidebarvsep

\sidebar의 ⟨*text*⟩ 인자는 \sidebarwidth 폭의 칼럼으로 식자된다. 그리고 본문 편집 영역과 ⟨*text*⟩가 놓일 영역 사이의 거리는 \sidebarhsep이다. \sidebarvsep 길이는 같은 페이지에 오는 사이드바들 사이의 수직 거리이다.

\setsidebarheight{⟨*length*⟩}

\setsidebarheight 매크로는 사이드바의 높이를 지정한다. ⟨*length*⟩ 인자는 행의 수를 나타내는 정수이다. 다음 예를 참고하라.

```
\setsidebarheight{15\onelineskip}
```

현재 사이드바 텍스트를 정렬하는 방식은 그다지 훌륭하지 않다. 사용자가 다양한 sidebarvsep과 \setsidebarheight를 이리저리 변경해가면서 더 좋은 값을 찾아내어야 할 것이다.[5]

5 만약 좀더 나은 정렬방식을 발견했다면 저자에게 알려주기 바란다.

13.4 사이드 노트

\marginpar로 만들어지는 사이드 노트의 수직 위치는 인접한 노트들이 겹치지 않도록 신축성있게 주어진다.

\sidepar[⟨*left*⟩]{⟨*right*⟩}
\sideparvshift

\sidepar 매크로는 \marginpar와 비슷하지만 사이드 노트가 플로트되지 않고 겹쳐 식자된다. \marginpar와 \sidepar는 동일한 \marginparsep과 \marinparwidth 길이를 사용한다. \sideparvshift 길이는 \sidepar 노트의 수직 위치를 조절할 때 사용된다. 기본값은 -2.08ex로서, 노트와 본문 행높이를 일치시키기 위해 선택된 값이다.

기본적으로 ⟨*right*⟩ 인자는 오른쪽 마진에 둔다. twoside 문서 옵션이 사용되면 ⟨*right*⟩ 인자는 짝수쪽 페이지에서는 왼쪽 마진에 온다. 그러나 ⟨*left*⟩ 마진이 주어지면 짝수쪽에서는 이 인자를 사용한다. 2단 문서에서는 단을 기준으로 '바깥쪽' 마진에 온다.

\sideparswitchtrue \sideparswitchfalse
\reversesidepartrue \reversesideparfalse
\parnopar

이 기본 위치는 \reversesidepar*와 \sideparswitch*선언을 적절하게 사용하여 바꿀 수 있다. twoside 문서에서 기본값은 \sideparswitchtrue이다. 이것은 마진 노트를 바깥쪽 마진에 두도록 하는 것이다. oneside 문서에서 기본값은 \sideparswitchfalse이다. 이것은 마진 노트를 언제나 오른쪽 마진에 두도록 한다. \reversesidepartrue 선언은 이 위치와는 반대로 되게 만든다. (기본값은 \reversesideparfalse이다.)

LaTeX이 사이드 노트를 어디에 둘 것인가를 결정할 때, 홀수쪽인지 짝수쪽인지를 검사한다. 그러나 막 페이지가 바뀐 경우 TeX이 홀짝수쪽을 잘 구분하지 못하는 경우가 있다. TeX은 (사이드 노트까지 포함하여) 한문단씩 조판하는데, 각 문단의 끝에 가서야 문단 중간에 페이지가 나누어졌는지를 알게 된다. 문단의 첫 부분이 출력되고, 페이지 나눔이 삽입되고 문단의 나머지 부분이 여전히 남아 있는 상태라면, 그것을 다시 배열하지 않는다면 새로운 페이지의 첫부분은 이전 문단으로 여전히 인식된다. 그래서 페이지가 바뀌었음에도 불구하고 여전히 이전 페이지의 마진을 이용함으로써 문단 끝에 있는 사이드 노트가 부적절한 위치에 놓이게 될 가능성이 있다. \parnopar 매크로는 새로운 패러그래프를 시작하게 하지만 겉으로 보기에는 그렇지 않게 하는 역할을 한다. 즉, 그 다음 문단의 첫 줄을 행바꿈 없이 이전 문단에 이어붙인다. \parnopar를 이용하여, TeX이 페이지 나눔을 계산할 때 하나의 문단을 두 개의 문단인 것처럼 생각하도록 할 수 있다.

Bastiaan Veelo는 사이드 노트의 또다른 형식의 예제 코드를 보내주었다. 그것은 다음과 같다.

```
%% A new command that allows you to note down ideas or annotations in
%% the margin of the draft. If you are printing on a stock that is wider
%% than the final page width, we will go to some length to utilise the
%% paper that would otherwise be trimmed away, assuming you will not be
%% trimming the draft. These notes will not be printed when we are not
%% in draft mode.
\makeatletter
   \ifdraftdoc
     \newlength{\draftnotewidth}
     \newlength{\draftnotesignwidth}
     \newcommand{\draftnote}[1]{\@bsphack%
       {%% do not interfere with settings for other marginal notes
         \strictpagechecktrue%
         \checkoddpage%
         \setlength{\draftnotewidth}{\foremargin}%
         \addtolength{\draftnotewidth}{\trimedge}%
         \addtolength{\draftnotewidth}{-3\marginparsep}%
         \ifoddpage
           \setlength{\marginparwidth}{\draftnotewidth}%
           \marginpar{\flushleft\textbf{\textit{\HUGE !\ }}\small #1}%
         \else
           \settowidth{\draftnotesignwidth}{\textbf{\textit{\HUGE\ !}}}%
           \addtolength{\draftnotewidth}{-\draftnotesignwidth}%
           \marginpar{\raggedleft\makebox[0pt][r]{%% hack around
               \parbox[t]{\draftnotewidth}{%%%%%%%%% funny behaviour
                 \raggedleft\small\hspace{0pt}#1%
               }}\textbf{\textit{\HUGE\ !}}%
           }%
         \fi
       }\@esphack}
   \else
     \newcommand{\draftnote}[1]{\@bsphack\@esphack}
   \fi
\makeatother
```

또 Bastiaan의 예제에서는 `\foremargin` 길이변수를 사용하고 있다. 이 코드를 시험해볼 생각이면, preamble에 두든가 별도의 패키지(즉, `.sty` 파일)로 만들어서 사용하면 된다. 별도의 패키지로 만들 때는 `\makeat...` 명령을 사용할 필요 없다.

제 14 장

표지(標識)와 장식

나의 영혼은 불멸의 삶을 찾는 것이 아니다. 오직 손 닿는 곳에 있는 원천을 만족시킬 즐거움을 찾는 것이다.

Pythian Odes
Pindar

이 장은 *companion* chapterstyle과 *companion* 페이지스타일로 작성되었다.

14.1 들어가는 말

지금까지 우리는 문서 조판의 여러 측면을 거의 모두 살펴보았다. 이 장에서는 이 클래수의 나머지 요소들, 즉 머릿글과 바닥글, 그리고 조금 더 재미있는 타입세팅 과제인 에피그라프에 대해서 알아본다.

14.2 페이지 스타일

이 클래스는 사용가능한 몇 가지 페이지 스타일을 제공한다. 만약 그것이 마음에 들지 않으면 스스로 정의해서 쓸 수도 있다.

【memhangul】 memhangul-ucs 패키지를 아무런 옵션 없이 쓰면 headings 스타일이 수정없이 적용된다. 그 때문에 '제 1 장'과 같은 표현이 면주에서 제대로 나오지 않을 수 있다. 이것은 오류가 아니며 의도된 것이므로 이를 피하려면 한글 문서에서는 예컨대 `\pagestyle{headings}`와 같이 특정 페이지 스타일을 preamble에서 별도로 지정해주는 것이 바람직하다. ■

이 기능은 fancyhdr [Oos96] 패키지에 영향을 받아 구현되었다. 그러나 사용되는 명령은 다르다.

표준 클래스는 짝수 및 홀수쪽의 머릿글과 바닥글을 제공한다. 그러므로 하나의 페이지 스타일을 정의하는 데 네 가지 요소가 필요하다. 이 클래스는 머릿글과 바닥글의 요소를 left, center, right 부분으로 나누므로, 모두 열두 개의 요소가 페이지 스타일에서 지정되어야 한다. 기정의 페이지 스타일이 대부분 무리없이 쓸 수 있을 것이므로 이 모든 정의에 대해서 크게 걱정하지 않아도 된다.

```
\pagestyle{⟨style⟩}
\thispagestyle{⟨style⟩}
```

`\pagestyle`은 현재의 페이지 스타일을 ⟨*style*⟩로 설정한다. 특정 페이지에서는 `\thispagestyle`을 사용하여 한 페이지만 현재 페이지 스타일을 오버라이드할 수 있다.

이 클래스가 `\thispagestyle`을 부르는 경우는 다음과 같다.

- `\part`는 `\thispagestyle{part}`를 부른다.
- `\chapter`는 `\thispagestyle{chapter}`를 부른다.
- 만약 `\cleardoublepage`가 호출되었을 때 왼쪽면을 비워야 하면 `\thispagestyle{cleared}`를 호출하여 빈 페이지를 만든다.

클래스에서 제공하는 페이지 스타일은 다음과 같다.

empty 머릿글과 바닥글에 아무것도 표시되지 않는다.

plain 머릿글은 비우고 페이지 번호는 페이지 바닥에 중앙정렬된다.

headings 바닥글을 비우고 머릿글에는 페이지의 바깥쪽에 페이지 번호와 장 또는 절의 표제를 붙인다. 제 5 장이 이 스타일을 사용하였다.

myheadings 이 스타일은 *headings*와 같지만 머릿글에 올 표제를 사용자가 지정할 수 있다.

ruled 바닥글의 바깥쪽[1]에 페이지 번호가 온다. 머릿글에는 장 또는 절의 표제를 넣고 머릿글 아래에 선을 긋는다. 이 스타일은 제 6 장에서 사용되었다.

Ruled 이것은 *ruled* 스타일과 같지만 머릿글과 바닥글이 여백까지 포함하여 만들어진다는 점이 다르다. 제 15 장이 이 스타일을 사용한다.

companion 이것은 *Companion* 시리즈(예를 들면 [GMS94])의 페이지 스타일과 비슷하다. 그 보기를 제 14 장에서 볼 수 있다.

part *plain* 페이지 스타일과 같다.

chapter *plain* 페이지 스타일과 같다.

cleared *empty* 페이지 스타일과 같다.

title *plain* 페이지 스타일과 같다.

titlingpage *empty* 페이지 스타일과 같다.

1 홀수쪽에서는 오른쪽, 짝수쪽에서는 왼쪽을 가리킨다.

headings 페이지 스타일은 머릿글에 들어갈 표제를 정의해주어야 한다. \sec과 같은 장절 명령은 각각 \secmark 매크로를 부르는데 페이지 스타일은 이 명령을 정의하여 거기에서 표제와 필요하다면 번호를 추출한다.

```
\markboth{⟨left⟩}{⟨right⟩}
\markright{⟨right⟩}
```

\markboth는 이 명령이 불리워진 위치에서 두 개의 *markers* 값을 각각 ⟨*left*⟩와 ⟨*right*⟩로 정의한다. 마찬가지로 \markright는 ⟨*right*⟩를 marker로 설정한다.

```
\leftmark \rightmark
```

\leftmark 매크로는 그 페이지의 마지막에 주어진 \markboth의 ⟨*left*⟩ 인자의 값을 기억하고 있다. \rightmark에는 그 페이지의 첫번째 \markboth 또는 \markright의 ⟨*right*⟩ 인자값을 저장하고 있다. 만약 이 값이 없으면 가장 최근의 ⟨*right*⟩ 값을 사용한다.

그래서 페이지 스타일은 \secmark 명령을 \markboth나 \markright를 이용하여 정의하며, \leftmark와 \rightmark를 머릿글이나 바닥글에 넣을 수 있게 된다. 나중에 이 일을 어떻게 하는지 예제를 보여주겠다. 이것은 *myheadings* 스타일이 구현되는 방법이기도 하다.

14.3 머릿글과 바닥글 만들기

앞서 말했듯이, 이 클래스는 짝수 및 홀수 쪽의 머릿글과 바닥글에 left, center, right 부분을 나누어두었다. 이 절에서는 이 열두 개의 부분을 사용하여 스스로의 페이지 스타일을 작성하는 방법을 설명한다.

```
\makepagestyle{⟨style⟩}
\aliaspagestyle{⟨alias⟩}{⟨original⟩}
\copypagestyle{⟨new⟩}{⟨original⟩}
```

\makepagestyle은 ⟨*style*⟩이라는 페이지 스타일을 설정한다. 기초값은 *empty* 페이지 스타일과 같다. \aliaspagestyle 매크로는 ⟨*alias*⟩ 페이지 스타일이 ⟨*original*⟩ 페이지 스타일과 같아지도록 한다. 이 패키지에는 다음과 같은 코드가 포함되어 있다.

```
\aliaspagestyle{part}{plain}
\aliaspagestyle{chapter}{plain}
\aliaspagestyle{cleared}{empty}
```

\copypagestyle은 ⟨*new*⟩ 페이지 스타일을 만들고 ⟨*original*⟩ 페이지 스타일 정의을 복사하여 온다.

```
\makeevenhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeevenfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
```

\makeevenhead 매크로는 짝수쪽(왼쪽면)의 페이지 스타일 머릿글 부분의 ⟨*left*⟩, ⟨*center*⟩, ⟨*right*⟩를 정의한다. \makeoddhead, \makeevenfoot, \makeoddfoot도 마찬가지로 각각 홀수쪽(오른쪽면)에 해당하는 ⟨*style*⟩의 머릿글 또는 바닥글의 각 부분을 정의한다. 이 명령들의 ⟨*style*⟩ 인자는 \makepagestyle의 ⟨*style*⟩에 해당하는 이름을 따라야 한다.

```
\makerunningwidth{⟨style⟩}{⟨length⟩}
\headwidth
```

\makerunningwidth 매크로는 ⟨*style*⟩ 페이지 스타일 머릿글과 바닥글의 너비가 ⟨*length*⟩가 되도록 설정한다. \makepagestyle은 이 값을 textwidth로 설정하므로, 이 매크로는 기본 너비와 다른 폭을 지정하고 싶을 때만 사용한다. \headwidth 길이변수는 머릿글과 바닥글 이외에도 다른 용도에 사용될 수 있는 길이 변수이다.

```
\makeheadrule{⟨style⟩}{⟨width⟩}{⟨thickness⟩}
\makefootrule{⟨style⟩}{⟨width⟩}{⟨thickness⟩}{⟨skip⟩}
```

머릿글 아래에는 선을 그을 수 있다. 그리고 바닥글의 위에도 선을 그을 수 있다. \makeheadrule 매크로는 ⟨*width*⟩와 ⟨*thickness*⟩ 값을 이용하여 머릿글과 본문 사이에 선을 긋는다. \makefootrule은 마찬가지로 본문과 바닥글 사이에 선을 긋는다. 이 매크로에 하나 더 있는 ⟨*skip*⟩ 인자는 foot rule의 수직 위치를 지정하기 위한 거리이다. \footruleskip을 보라. \makepagestyle 매크로는 ⟨*width*⟩ 값을 \textwidth로, ⟨*thickness*⟩를 0pt로 초기화하기 때문에 선이 나타나지 않는 것이 기본값이다.

```
\normalrulethickness
```

\normalrulethickness는 선의 기본 두께인데 0.4pt이다. \setlength로 바꿀 수 있기는 하지만 14pt나 17pt 옵션을 쓰는 경우가 아니라면 되도록 바꾸지 말 것을 권고한다.

```
\footruleheight
\footruleskip
```

\footruleheight 매크로는 바닥글 위에 긋는 선의 기본값이다(default는 0). \footruleskip은 편집영역의 아래쪽에서 바닥글 사이에서 선이 그어질 위치를 확보하기 위한 거리이다. 이 값은 길이 변수 같이 보이지만 바꾸고 싶다면 \setlength가 아닌 \renewcommand를 써야 한다.

\makeheadposition{⟨*style*⟩}
{⟨*eheadpos*⟩}{⟨*oheadpos*⟩}{⟨*efootpos*⟩}{⟨*ofootpos*⟩}

\makeheadposition 매크로는 각각, ⟨*style*⟩에 해당하는 홀수 및 짝수쪽 머릿글과 바닥글의 수평 위치를 지정한다. ⟨*...pos*⟩ 인자에 올 수 있는 것은 flushleft, center, flushright이다. 인자가 주어지지 않거나 받아들일 수 없으면 center로 간주한다. 이 매크로는 머릿글 바닥글 폭이 \textwidth와 같지 않을 때만 유용하다.

\makepsmarks{⟨*style*⟩}{⟨*code*⟩}

\pagestyle{⟨*style*⟩}이 하는 마지막 일은 ⟨*style*⟩에 해당하는 \makepsmarks의 ⟨*code*⟩ 인자를 부르는 것이다. 보통 비표준 코드를 지정하고 싶을 때(즉, 앞서 나온 매크로를 이용해서는 지정할 수 없는 코드를 사용할 때) 사용할 수 있다.

페이지 스타일 예제

이 첫번째 예제는 페이지 스타일 정의 명령 대부분을 시험해본다. *LaTeX Companion* 시리즈 [GMS94, GRM97, GR99]는 머릿글이 편집 영역보다 넓고 여백까지 먹어들어가 있다. 그리고 그 아래 선을 그었다. 페이지 번호는 머릿글의 바깥쪽에 bold로 식자하였다. 장 표제를 왼쪽 페이지 머릿글에 넣었고 오른쪽 페이지 머릿글에는 절표제를, 둘 다 bold으로 안쪽 마진에 두었다. 바닥글은 비어 있다.

이 스타일을 구현하려 할 때 첫번째로 해야 할 일은 머릿글의 폭을 계산하는 것이다. 이 폭이 마진 노트 영역까지 연장되어야 한다.

```
\setlength{\headwidth}{\textwidth}
  \addtolength{\headwidth}{\marginparsep}
  \addtolength{\headwidth}{\marginparwidth}
```

먼저 *empty* 페이지 스타일을 정의하고 그것을 수정하여 새로운 머릿글 바닥글의 폭을 지정하여 보자.

```
\makepagestyle{companion}
\makerunningwidth{companion}{\headwidth}
```

그리고 헤더 선의 폭과 두께를 지정해야 한다. 하지 않으면 선이 나타나지 않는다.

```
\makeheadrule{companion}{\headwidth}{\normalrulethickness}
```

머릿글이 바깥쪽 마진으로 나가도록 짝수쪽 머릿글은 flushright(raggedleft)하고 홀수쪽 머릿글은 flushleft(raggedright)하자. 바닥글은 비울 것이므로 신경쓰지 않는다.

```
\makeheadposition{companion}{flushright}{flushleft}{}{}
```

현재 장과 절의 표제는 \leftmark와 \rightmark 매크로로 얻는다. 이들은 \chaptermark와 \sectionmark 매크로를 통하여 정의된다. \leftmark는 그 페이지의 마지막 ⟨*left*⟩ marker이고 \rightmark는 첫번째 ⟨*right*⟩ marker라는 사실을 기억하자.

장 번호는 머릿글에 두지 않지만 절 번호가 붙어 있다면 머릿글에 함께 둔다. 올바른 정의가 이루어지도록 하려면 주의를 기울여야 하는데, 이 대목이 \makepsmarks 매크로가 역할을 하는 곳이다.

```
\makepsmarks{companion}{%
  \let\@mkboth\markboth
  \def\chaptermark##1{\markboth{##1}{##1}}% % left & right marks
  \def\sectionmark##1{\markright{%          % right mark
    \ifnum \c@secnumdepth>\z@
      \thesection. \%                      % section number
    \fi
    ##1}}
}
```

기본적인 정의는 완성되었다. 이제 남은 것은 각각의 머릿글과 바닥글 각 부분에 어떤 내용을 넣는가를 지정하는 것이다. 바닥글은 비워야 한다.

```
\makeevenhead{companion}%
  {\normalfont\bfseries\thepage}{}{\normalfont\bfseries\leftmark}
\makeoddhead{companion}%
  {\normalfont\bfseries\rightmark}{}{\normalfont\bfseries\thepage}
```

이제 pagestyle{companion} 명령을 주어서 페이지가 *companion* 페이지 스타일 머릿글로 만들어지도록 하자.

이상의 정의는 실제로 이 클래스의 일부분이다. 만약 이 설정을 바꾸려 할 때나 이를 기초로 자신의 스타일을 설정할 때는 @ 문자가 들어 있는 매크로의 이름에 주의하라.

특별한 머릿글

찾아보기를 보면 머릿글이 그 페이지의 첫번째와 마지막 엔트리를 보여주고 있음을 볼 수 있을 것이다. 인덱스의 메인 엔트리는 다음과 같다.

```
\item \idxmark{entry}, page number(s)
```

그리고 이 안내서의 preamble에 `\idxmark`를 다음과 같이 정의했다.

```
\newcommand{\idxmark}[1]{#1\markboth{#1}{#1}}
```

이것은 엔트리를 식자하고 동시에 marker로 사용하여 그 페이지의 첫번째 엔트리가 `\rightmark`, 마지막 엔트리가 `\leftmark`가 되도록 한 것이다.

색인 엔트리는 대개 매우 짧기 때문에 찾아보기는 2단으로 조반된다. 불행히도 LaTeX의 marking 메카니즘은 약간 불안정해서 2단 조판 페이지에서는 풀리는 경우가 있다. 그러나 fixltx2e [MC00] 표준 패키지는 이것을 수정해준다.

색인 자체는 다음과 같이 부른다.

```
\clearpage
\pagestyle{index}
\renewcommand{\preindexhook}{%
The first page number is usually, but not always, the primary reference to
the indexed topic.\vskip\onelineskip}
\printindex
```

index 페이지 스타일은 이 예제의 핵심이다. 다음과 같이 정의하였다.

```
\makepagestyle{index}
  \makeheadrule{index}{\textwidth}{\normalrulethickness}
  \makeevenhead{index}{\rightmark}{}{\leftmark}
  \makeoddhead{index}{\rightmark}{}{\leftmark}
  \makeevenfoot{index}{\thepage}{}{}
  \makeoddfoot{index}{}{}{\thepage}
```

아마도 알아차렸을 것으로 보지만, 여기서는 찾아보기의 첫번째와 마지막 엔트리를 머릿글의 왼쪽과 오른쪽에 둔다. 페이지 바닥의 바깥쪽 여백에 페이지 번호를 둔다.

Pehong Chen의 *MakeIndex* 프로그램 [CH88]을 이용하여 `.idx` 파일에 들어 있는 인덱스 데이터를 LaTeX이 기대하는 형식인 `.ind` 파일로 변환하였다. *MakeIndex* 프로그램은 `.ist` 파일을 이용하여 동작을 제어할 수 있는데, 이 안내서를 위해서 `memman.ist` 파일을 만들었고, 여기에는 다음과 같은 희한한 내용이 적혀 있다.

```
% output main entry <entry> as: \item \idxmark{<entry>},
item_0  "\n\\item \\idxmark{"
delim_0 "},"
```

이 암호같은 코드에 대한 설명을 보려면 Chen의 논문 [CH88]이나 [GMS94]의 제12장을 참고하라.

`\ifonlyfloats{`⟨*yes*⟩`}{`⟨*no*⟩`}`

그림이나 표만을 포함하고 있는 페이지의 헤더를 보통 페이지와 다르게 하는 것이 필요한 경우가 있다. 텍스트 없이 플로트만을 포함한 어떤 페이지에서 `\ifonlyfloats` 명령이 내려지면 ⟨*yes*⟩ 인자가 실행된다. 만약 보통 페이지에서라면 ⟨*no*⟩ 인자가 처리된다. 이 명령은 플로트만을 포함한 페이지를 다른 모양으로 설정하는 페이지 스타일을 정의할 때 특히 유용하다.

예를 들자면, *companion* 페이지 스타일을 일반적으로 사용하면서 플로트만으로 된 페이지는 *plain*에 가깝게 만들려 한다고 가정해보자. *companion* 설정 코드에서 몇 가지를 가져와서 다음과 같이 하면 된다.

```
\makepagestyle{floatcomp}
% \headwidth has already been defined for the companion style
\makeheadrule{floatcomp}{\headwidth}%
  {\ifonlyfloats{0pt}{\normalrulethickness}}
\makeheadposition{floatcomp}{flushright}{flushleft}{}{}
\makepsmarks{floatcomp}{\companionpshook}
\makeevenhead{floatcomp}{\ifonlyfloats{}{\normalfont\bfseries\thepage}}%
           {}{\ifonlyfloats{}{\normalfont\bfseries\leftmark}}
\makeoddhead{floatcomp}{\ifonlyfloats{}{\normalfont\bfseries\rightmark}}%
           {}{\ifonlyfloats{}{\normalfont\bfseries\thepage}}
\makeevenfoot{floatcomp}{}{\ifonlyfloats{\thepage}{}}{}
\makeoddfoot{floatcomp}{}{\ifonlyfloats{\thepage}{}}{}
```

위의 코드 *floatcomp* 스타일과 앞서 정의한 *companion* 스타일을 비교해보라.

플로트 페이지에서는 머릿글의 선 두께를 0으로 만들어서 보이지 않게 한다. 일반적으로는 `\normalrulethickness`이다. 머릿글의 위치는 두가지 페이지 스타일에서 동일하다. 그러나 *floatcomp*에서는 플로트 페이지의 헤더가 비어 있고 바닥글은 페이지 번호가 중앙정렬된다. 보통 페이지에서는 바닥글이 비고 헤더는 *companion* 헤더와 동일하다.

이 코드에는 한 가지 '트릭'이 사용되었다. `\makepsmarks{X}{code}` 매크로는 다음처럼 하는 것과 같다.

```
\newcommand{\Xpshook}{code}
```

여기서 이것을 적용하였다.

```
\makepsmarks{floatcomp}{\companionpshook}
```

이렇게 하면 `\makepsmarks{companion}{...}`에 사용했던 코드를 다시 쓰지 않아도 된다. 그리고 두 가지 페이지 스타일에 실제로 동일한 코드를 적용할 수 있다.

`\mergepagefloatstyle{`⟨*style*⟩`}{`⟨*textstyle*⟩`}{`⟨*floatstyle*⟩`}`

이미 정의되어 있는 두 가지 페이지 스타일을 하나는 보통 텍스트 페이지에, 다른 하나는 플로트 페이지에 적용하고 싶은 경우를 생각해보자. `\mergepagefloatstyle` 명령을 사용하면 위에서 보았던 *floatcomp* 코드 예제보다 더 쉽게 이 둘을 결합할 수 있다. ⟨*style*⟩은 정의하려 하는 스타일의 이름이다. ⟨*textstyle*⟩은 텍스트 페이지의 페이지 스타일 이름이고, ⟨*floatstyle*⟩은 플로트 페이지에 적용할 페이지 스타일 이름이다. 이 두 개의 페이지 스타일은 `\mergepagefloatstyle` 명령이 불리기 전에 정의되어 있어야 한다. 따라서, 위에서처럼 길고 복잡하게 정의할 것 없이, 다음과 같이 하는 것이 가능하다.

```
\mergepagefloatstyle{floatcomp}{companion}{plain}
```

14.4 에피그라프

> 전체는 부분보다 크다.
>
> *Metaphysica*
> *Aristotle*

재미있는 인용문을 장절의 처음이나 끝에 두고 싶어하는 저자들이 있다. 이 클래스는 한 개의 에피그라프 조판을 지원한다. 또 어떤 저자들은 여러 개의 인용구들을 두고 싶어한다. 그래서 이 클래스는 이것을 가능하게 하는 환경도 제공한다. 에피그라프는 편집 영역의 왼쪽, 중앙, 오른쪽 어디든지 둘 수 있다. 몇 가지 에피그라프의 예를 여기서 보였는데, Christina Thiele [Thi99]이 작성한 epigraph 패키지 [Wil00a]에 대해 쓴 글을 읽어보면 더 많은 보기를 찾을 수 있을 것이다. 이 패키지는 이 클래스에 포함되어 있다.

epigraph 명령

`\epigraph` 명령의 맨 처음 시사는 우리의 책 [SW94]에서 Doug Schenck가 제시한 아이디어였다. 그 목적을 구현하는 것은 힘든 일이었지만, 이제는 훨씬 신축적으로 이 기능을 제공하게 되었다.

```
\epigraph{⟨text⟩}{⟨source⟩}
```

epigraph 명령은 ⟨*text*⟩를 에피그라프 텍스트로 하고 ⟨*source*⟩를 원저자(또는 책, 논문 등)로 하는 에피그라프를 식자한다. 에피그라프는 편집 영역의 오른쪽에 놓이고 ⟨*source*⟩가 ⟨*text*⟩의 오른쪽 아래에 오도록 하는 것이 기본값이다.

epigraphs 환경

```
\begin{epigraphs}
\qitem{⟨text⟩}{⟨source⟩}
...
\end{epigraphs}
```

epigraphs 환경은 여러 개의 에피그라프를 식자한다. 기본값은 편집 영역의 오른쪽에 오는 것이다. 각 에피그라프들은 \qitem(일반적인 리스트 환경의 \item와 유사하다)으로 지정한다. ⟨*source*⟩는 ⟨*text*⟩의 오른쪽 아래에 두는 것이 기본값이다.

일반적 용법

> Example is the school of mankind, and they will learn at no other.
>
> *Letters on a Regicide Peace*
> Edmund Burke

여기에서 설명하는 명령들은 \epigraph 명령과 epigraphs 환경에 공통으로 적용된다. 먼저 지적해둘 것은, 장절 헤딩에 바로 이어서 에피그라프가 오면 첫번째 문단은 들여쓰기가 될 것이라는 점이다. 첫 문단이 들여쓰기 되지 않게 하려면 그 문단을 \noindent 명령으로 시작해야 한다.

```
\epigraphwidth
\epigraphtextposition{⟨flush⟩}
```

에피그라프는 \epigraphwidth 폭을 갖는 minipage에 식자된다. 이 값은 \setlength 명령으로 바꿀 수 있다. 일반적으로 에피그라프의 문단폭은 편집 영역보다 훨씬 적은 값이다. 행나눔의 혼란을 방지하기 위해서 ⟨*text*⟩는 raggedright로 조판하는 것이 보통이다.

\epigraphtextposition 선언의 ⟨*flush*⟩ 인자는 ⟨*text*⟩의 조판 형식을 조절한다. 기본값은 flushleft 즉 raggedright이다. 여기에 올 수 있는 그밖의 값은 center, flushright, flushleftright들이며, 각각 중앙정렬, 오른쪽정렬, 양쪽정렬로 조판된다.

어쩌다가 ⟨*text*⟩가 별스러운 레이아웃 스타일로 조판되기를 바란다면, 가장 쉬운 방법은 문단모양을 정의하는 파라미터를 재설정하는 새로운 환경을 정의하는 것이다. 예컨대, ⟨*text*⟩가 minipage 안에서는 들여쓰기되지 않는다. 이것을 들여쓰도록 하고 양쪽정렬로 만들고 싶다면 다음과 같은 새로운 환경을 정의한다.

```
\newenvironment{myparastyle}{\setlength{\parindent}{1em}}{}
```

이것을 다음과 같이 사용한다.

```
\epigraphtextposition{myparastyle}
```

`\epigraphposition{`⟨*flush*⟩`}`

앞서 말했듯이, 에피그라프의 기본 위치는 편집 영역의 오른쪽이다. 이 위치를 조절하는 데는 `\epigraphposition` 선언의 ⟨*flush*⟩ 인자가 쓰인다. 기본값은 `flushright`이다. 여기에 올 수 있는 설정 인자는 이외에 `flushleft`와 `center`가 있다.

`\epigraphsourceposition{`⟨*flush*⟩`}`

`\epigraphsourceposition` 선언의 ⟨*flush*⟩ 인자는 ⟨*source*⟩의 위치를 결정한다. 기본값은 `flushright`이다. 여기에 올 수 있는 설정 인자는 `flushleft`, `center`, `flushleftright`가 더 있다.

에피그라프는 중앙정렬하고 ⟨*source*⟩는 왼쪽에 두려면 다음과 같이 지정하면 된다.

```
\epigraphposition{center}
\epigraphsourceposition{flushleft}
```

`\epigraphfontsize{`⟨*fontsize*⟩`}`

에피그라프는 보통 본텍스트보다 조금 작은 활자로 인쇄된다. `\epigraphfontsize` 선언의 ⟨*fontsize*⟩ 인자가 이 글꼴 크기를 설정하는 데 쓰인다. 기본 글꼴 크기(`\small`)가 마음에 들지 않으면 쉽게 바꿀 수 있다.

```
\epigraphfontsize{\footnotesize}
```

`\epigraphrule`

⟨*text*⟩와 ⟨*source*⟩ 사이에 줄을 긋는 것이 기본값이다. 줄의 굵기는 `\epigraphrule` 값에 의해 주어지고, 이 값은 `\setlength`를 통해 바꿀 수 있다. 이 값을 `0pt`로 하면 줄이 없어진다. 개인적으로 list 환경에 줄을 긋는 것을 싫어하는 편이다.

```
\beforeepigraphskip
\afterepigraphskip
```

두 개의 `...skip` 명령은 에피그라프를 인쇄하기 전후의 간격을 설정한다. 역시 `\setlength`로 변경할 수 있다. 이 값의 합이 `\baselineskip`의 정수배가 되도록 하는 것이 좋다.

⟨*text*⟩나 ⟨*source*⟩ 인자에 일반적인 LaTeX 명령을 사용할 수 있다. 즉, ⟨*text*⟩를 로마체로 하고 ⟨*source*⟩를 이탤릭체로 하는 것도 가능하다.

이 소절의 시작 부분에 있는 에피그라프는 다음과 같이 코딩한 것이다.

```
\epigraph{Example is the school of mankind,
          and they will learn at no other.}
 {\textit{Letters on a Regicide Peace}\\ \textsc{Edmund Burke}}
```

chapter 헤딩 앞에 두는 에피그라프

`\epigraph` 명령과 `epigraphs` 환경은 이 명령이나 환경이 불린 위치에 에피그라프를 식자한다. 그런데 `\chapter` 명령이 제일 처음 하는 일은 새로운 페이지를 시작하는 것이므로, 에피그라프를 chapter heading 직전에 두려면 별도의 메카니즘이 필요하다.

```
\epigraphhead[⟨distance⟩]{⟨text⟩}
```

`\epigraphhead` 매크로는 ⟨*text*⟩를 저장했다가 그 페이지의 헤더 아래 ⟨*distance*⟩만큼 거리를 두고 인쇄한다. ⟨*text*⟩는 일반적인 텍스트이거나, `\epigraph` 명령 또는 `epigraphs` 환경일 수 있다. 에피그라프는 오른쪽 마진에 붙여서 인쇄되는 것이 기본이다. 만약 이 명령이 `\chapter`나 `\chapter*` 명령 직전에 주어지면 에피그라프는 장 타이틀 페이지에 식자된다.

⟨*distance*⟩ 선택 인자의 기본값은 `\epigraph`가 한 줄의 인용문과 한 줄의 source로 이루어졌을 때, `\chapter` 명령에 의하여 만들어지는 'Chapter X' 행의 바닥에 가지런하게 정렬되도록 해준다. 이 값은 정수이거나 소수값일 수 있다. 단위는 현재 설정되어 있는 `\unitlength`이다. 한 줄짜리 인용문과 한 줄짜리 소스를 가진 에피그라프를 `\chapter*`에 붙이는 일반적인 길이값은 약 70(포인트) 정도이다. ⟨*distance*⟩가 양수값일 때 에피그라프는 페이지 헤딩에서 아래로 식자되고 음수값일 때는 페이지 헤딩 위쪽으로 간다.

```
\chapter*{Celestial navigation}
\epigraphhead[70]{\epigraph{Star crossed lovers.}{\textit{The Bard}}}
```

⟨*text*⟩인자는 \epigraphwidth의 폭을 가지는 minipage 안에 넣는다. 만약 \epigraph나 epigraphs가 아닌 다른 것을 ⟨*text*⟩ 인자로 주면 텍스트의 위치는 사용자가 직접 적절하게 조절해야 한다. 예를 들면,

```
\chapter{Short}
\renewcommand{\epigraphflush}{center}
\epigraphhead{\centerline{Short quote}}
```

\epigraphhead 명령은 에피그라프가 위치할 페이지 스타일을 바꾼다. 그러므로 \chapter와 \epigraphhead 명령 사이에는 텍스트가 오지 않아야 한다. 이 페이지 스타일은 *plain*과 같지만 에피그라프를 포함한다는 것만이 다르다. 에피그라프가 붙은 장의 페이지 스타일을 좀더 멋지게 하고 싶으면 스스로 설정을 해보도록 하라.

\epigraphforheader[⟨*distance*⟩]{⟨*text*⟩}
\epigraphpicture

\epigraphforhead 매크로는 \epigraphhead와 동일한 인자를 취하지만 ⟨*text*⟩를 좌표 위치 (0,-<distance>에 사이즈가 0인 그림으로 위치짓는다. \epigraphpicture 매크로는 이 그림을 저장하고 있다. 이들은 그 자체로 chapter 페이지 스타일의 일부로 이용될 수 있다. 예를 들면 다음과 같다.

```
\makepagestyle{mychapterpagestyle}
...
\makeoddhead{mychapterpagestyle}{}{}{\epigraphpicture}
```

\dropchapter{⟨*length*⟩}
\undodrop

긴 에피그라프가 장 타이틀 이전에 오게 되면 에피그라프의 마지막이 장 타이틀을 교락할 것이다. \dropchapter 명령은 에피그라프 뒤에 이어지는 장 타이틀을 ⟨*length*⟩ 길이만큼 아래로 내려준다. ⟨*length*⟩ 값이 음수이면 장 타이틀이 위로 이동한다. \undodrop 명령은 그 이후의 장 타이틀을 원래의 위치로 되돌려준다. 아래는 보기이다.

```
\dropchapter{2in}
\chapter{Title}
\epigraphhead{long epigraph}
\undodrop
```

`\cleartoevenpage[`⟨*text*⟩`]`

이따금 장이 시작하는 페이지의 맞은편에 어떤 내용, 예컨대 에피그라프나 지도, 그림 등을 두고자 할 때가 있다. 이 때 그 내용은 새로 시작될 장에 속하는 것이지 이전에 끝난 장에 해당하는 내용이 아니다. 이렇게 하려면 문서가 양면(doublesided)으로 작성되어야 하고, chapter는 반드시 홀수쪽에서 시작하고 이전 장의 내용이 짝수쪽 페이지 직전에서 끝나야 한다. `\cleartoevenpage` 명령은 `\cleardoublepage`와 비슷하지만 그 다음 페이지가 짝수쪽이 되도록 하는 역할을 한다. 그리고 필요하다면 선택 인자를 주어서 건너뛸 페이지가 몇 페이지인가를 지정할 수도 있다.

예를 들어보자.

```
... end previous chapter.
\cleartoevenpage
\begin{center}
\begin{picture}... \end{picture}
\end{center}
\chapter{Next chapter}
```

만약 페이지의 상단에 앞선 장의 헤딩이 찍힌다면, `\thispagestyle{empty}` 또는 `plain`을 `\cleartoevenpage` 명령 직전에 두는 것이 좋을 것이다.

만약 번호붙은 캡션이 있는 그림 같은 것이 있고 번호가 장 번호에 따른다면, 번호는 직접 수정해주어야 한다(특별한 chapter command를 따로 정의하지 않는 한). 다음은 보기이다.

```
... end previous chapter.
\cleartoevenpage[\thispagestyle{empty}] % skipped page, if any, to be empty
\thispagestyle{plain}
\addtocounter{chapter}{1} % increment the chapter number
\setcounter{figure}{0}    % initialise figure counter
\begin{figure}
...
\caption{Pre chapter figure}
\end{figure}

\addtocounter{chapter}{-1} % decrement the chapter number
\chapter{Next chapter}     % increments chapter & initialises figure numbers
\addtocounter{figure}{1}   % to account for pre-chapter figure
```

Part 페이지의 에피그라프

epigraph 패키지에서는 book이나 report 클래스의 \part 또는 \part*와 같은 페이지에는 에피그라프를 둘 수 없는 듯이 보인다. 그 이유는 \Part 명령이 내부적으로 타이틀 페이지 전후에 페이지 플리핑을 하기 때문이다. 그러나 part 페이지에 에피그라프를 두는 것은 아주 쉽다.

- epipart.sty라는 파일을 다음 내용으로 작성한다.
```
% epipart.sty
\let\@epipart\@endpart
\renewcommand{\@endpart}{\thispagestyle{epigraph}\@epipart}
\endinput
```

- 문서 시작 부분에 다음과 같이 써둔다.
```
\documentclass[...]{...}
\usepackage{epigraph}
\usepackage{epipart}
```

- \part 명령의 직전에 \epigraphhead 명령을 둔다.
```
\epigraphhead[300]{Epigraph text}
\part{Part title}
```

  옵션 인자는 에피그라프의 수직 위치를 조정하는 데 쓰인다. \part 중에서 에피그라프를 갖지 않는 것이 있으면 \epigraphhead 명령을 비워준다. 즉, \epigraphhead{}를 \part 명령의 직전에 둔다.

비슷한 방법으로 다른 페이지에도 에피그라프를 둘 수 있다. 핵심적인 트릭은 그 페이지에 *epigraph* 페이지 스타일을 설정하는 것이다. 문헌 목록이나 찾아보기에 에피그라프를 두고 싶다면 \prebibhook 또는 \preindexhook 매크로를 이용한다.

십오 장

운문 조판

이 장은 *demo* 장 스타일과 *Ruled* 페이지스타일을 사용한다.

15.1 개요

시(詩)를 조판하는 것은 그 시가 어떠하냐에 전적으로 달려 있다. 각각의 시가 가진 특성을 구현하는 것은 일반적 해결책으로 될 일이 아니기 때문에, 이 사용안내서는 특정한 시 작품을 조판하는 일반화된 해결책을 제시하려는 것이 아니라 해결책을 구할 수 있는 방법을 안내하는 데 그치려고 한다.

설명을 위해 사용한 날림 시는 [Wil??]에서 가져왔다.

이 절에서 사용된 예제들은 최소한의 (LA)TEX 기능 이상의 것을 사용하지 않았다는 점에 주의하라. 특히, 몇 연이 건너편 페이지로 왔다갔다한다든가 하는 특별한 페이지 나누기 — 출판인들이 전혀 반기지 않는 — 를 적용하려는 시도 등을 하지 않았다.

표준 LATEX 클래스들은 `verse`라는 특별한 리스트 환경을 제공한다. 이 환경에서는 `\\`로 행끝을 표시하고 연의 끝은 빈 줄을 둔다. 다음은 간단한 1연 시이다.

```
\newcommand{\garden}{
I used to love my garden \\
But now my love is dead \\
For I found a bachelor's button \\
In black-eyed Susan's bed.
}
```

이 내용을 LATEX의 일반적 문단으로 조판하면 다음과 같이 된다. 문단 들여쓰기는 적용하지 않았다.

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

이것을 verse 환경으로 조판하면 다음과 같이 된다.

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

이 한 연을 alltt 환경으로 조판할 수도 있다. 이 환경은 표준 alltt 패키지 [Bra97] 에 정의되어 있다.

```
\begin{alltt}\normalfont
I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.
\end{alltt}
```

결과는 다음과 같다.

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

alltt 환경은 verbatim 환경과 같지만 LaTeX 매크로를 그 안에서 사용할 수 있다. verse 환경은 긴 줄을 개행하고 들여쓰기하지만 alltt 환경은 들여쓰기가 없다.

한 연 안에서 몇 행이 들여쓰기되는 경우가 있다. 이런 연을 조판하려면 들여쓰기 공백을 직접 지시해주어야 한다. 예를 들어보자.

```
\begin{verse}
There was an old party of Lyme \\
Who married three wives at one time. \\
\hspace{2em} When asked: 'Why the third?' \\
\hspace{2em} He replied: 'One's absurd, \\
And bigamy, sir, is a crime.'
```

```
\end{verse}
```

이것은 다음과 같이 나타난다.

There was an old party of Lyme
Who married three wives at one time.
When asked: ‘Why the third?’
He replied: ‘One’s absurd,
And bigamy, sir, is a crime.’

`alltt` 환경을 사용하면 공백을 그냥 넣을 수 있다.

```
\begin{alltt}\normalfont
There was an old party of Lyme
Who married three wives at one time.
      When asked: ‘Why the third?’
      He replied: ‘One’s absurd,
And bigamy, sir, is a crime.’
\end{alltt}
```

결과는 다음과 같다.

```
There was an old party of Lyme
Who married three wives at one time.
      When asked: ‘Why the third?’
      He replied: ‘One’s absurd,
And bigamy, sir, is a crime.’
```

더 재미있는 것은 TEX의 `\parshape` 명령[1]을 이용하는 것이다.

```
\parshape = 5 0pt \linewidth 0pt \linewidth
             2em \linewidth 2em \linewidth 0pt \linewidth
\noindent There was an old party of Lyme \\
Who married three wives at one time. \\
When asked: ‘Why the third?’ \\
He replied: ‘One’s absurd, \\
And bigamy, sir, is a crime.’ \par
```

1 사용법은 *TEXbook*을 보라

이것은 다음과 같이 나타난다.

There was an old party of Lyme
Who married three wives at one time.
When asked: 'Why the third?'
He replied: 'One's absurd,
And bigamy, sir, is a crime.'

이 정도가 표준 (LA)TEX에서 제공하는 기능이다. 다만, verse 환경 내에서 *와 같이 별표붙은 \\를 쓰면 그 다음 줄에서 페이지 나누기를 금지하는 효과가 있다는 정도를 알아두자. \needspace 매크로를 적절하게 사용하여도 같은 일을 할 수 있다.

시집 특히 시선집은 두 가지 이상의 색인을 가지기도 한다. 하나는 시 제목 색인이고 두번째 것은 첫행 색인이다. index 패키지[Jon95]와 multind 패키지[Lon91]는 다중 색인을 지원한다.

15.2 memoir 클래스의 고급 운문 조판

【memhangul】 이 절의 제목은 classy verse이다. 이 표현은 이중적 의미를 가지고 있는데, 한편으로는 "걸작에 속하는" 시라는 뜻이기도 하고, 다른 한편으로는 이 "클래스의" 운문 조판 기능을 가리키기도 한다. 우리말로는 이런 이중적 의미를 모두 표현할 단어를 찾을 수 없어서 위와 같이 번역하였다. ■

memoir 클래스가 제공하는 것은 시 조판에 필요한 몇 가지 기능들일 뿐이지, 가능한 모든 경우에 대응하는 완벽한 해결책을 제시하려는 것은 아니고, 가능하지도 않다.

일반적인 텍스트를 조판할 때와 시 조판의 차이점 중 주요한 것을 들어보면,

- 시는 대체로 시각적으로 페이지 중앙에 온다.
- 들여쓰기가 이루어지는 줄이 있다. 이따금 들여쓰기가 반복해서 같은 패턴으로 이루어지도 한다.
- 행이 페이지 폭에 비해 너무 길면 나누어서 나머지 부분을 원래 줄보다 안쪽으로 일정 비율 들여쓴다.

이 클래스가 구현하고 있는 것은 아래와 같다.

```
\begin{verse}[⟨length⟩] ... \end{verse}
\versewidth
\stanzaskip
```

이 클래스가 제공하는 verse 환경은 표준 LATEX 환경을 확장한 것이다. 길이값 옵션 인자 하나를, 예를 들면 \begin{verse}[4em]와 같이 줄 수 있다. 앞서 본 대로 verse 예제들은

왼쪽 마진에 붙여서 식자되고 있다. 그러나 일반적으로 페이지의 중앙에 배치되는 것이 더 보기 좋다. 이 길이값 인자가 주어지면 이것을 평균적인 행의 길이로 보아서 그것을 기준으로 페이지의 중앙에 전체 내용이 놓이도록 조판한다.

길이변수 \versewidth를 이용하면 편리하다. 예를 들면 이 명령은 인자로 주어지는 문자열의 길이를 계산하여 그 값을 취하므로, 나중에 verse 환경에서 옵션인자로 지정할 수 있다.

```
\settowidth{\versewidth}{This is the average line,}
\begin{verse}[\versewidth]
```

연과 연 사이의 수직 간격은 \stanzaskip 길이변수로 설정되므로, 길이변수를 수정하는 방법을 적용하여 바꿀 수 있다.

```
\\ \\* \\> \\!
```

verse 환경 내의 모든 행은 마지막 행만을 제외하고 모두 \\를 행끝에 두어야 한다. 별표 붙인 *는 다음 줄과의 사이에서 페이지를 나누지 못하게 억제한다. \\>는 행 중간에서 행나눔을 강제할 때 쓴다. \verselinebreak 매크로의 설명을 참고하라.

\\! 매크로는 행끝에 와서 연의 끝이기도 함을 나타낸다. 이 명령은 다음 연과의 사이에 빈 줄을 하나 넣어준다.

\\... 매크로들은 모두 하나의 길이값 옵션 인자를 취한다. \\, *, \\! 매크로의 경우 옵션 인자 길이값은 행간의 수직 간격을 지정한다. \\> 매크로의 경우에는 수평 간격이다.

```
\vin
\vgap
\vindent
```

\vin 명령은 \hspace*{\vgap}의 축약형인데, 시행 가운데 들여쓰기하는 행의 첫머리에 사용한다. \vgap 길이(초기값은 1.5em)는 \setlength나 \addtolength로 바꿀 수 있다.

시행들은 이따금 이전 행 텍스트 길이에 맞추어서 들여써야 할 때가 있다.

```
\vinphantom{⟨text⟩}
```

\vinphantom 매크로는 행의 첫머리에 쓰여서 ⟨*text*⟩ 만큼 들여쓰기 행이 되도록 해준다. 예를 들면 다음과 같다.

...
Come away with me.

Impossible!

...

위의 각 부분은 다음과 같이 입력된 것이다.

```
\begin{verse}
\ldots \\
Come away with me.
\end{verse}
\begin{verse}
\vinphantom{Come away with me.} Impossible! \\
\ldots
\end{verse}
```

\vinphantom은 행의 중간에서 일정 크기의 공백을 남기려 할 때도 쓸 수 있다. 다음 코드에 의해서 만들어지는 두 행을 비교해보라.

```
\noindent Come away with me and be my love --- Impossible. \\
          Come away with me \vinphantom{and be my love} --- Impossible.
```

Come away with me and be my love — Impossible.
Come away with me — Impossible.

시의 한 행이 너무 길어서 편집영역에 들어가지 못하게 되면 약간의 공간을 두면서 다음 줄로 나누어진다. 이 공간의 크기는 \vindent 길이로 주어진다. 기본값은 \vgap의 두 배이다.

\verselinebreak[⟨*length*⟩]

\verselinebreak 명령을 사용하면 시가 조판된 이후의 행에서도 다음 줄이 들여쓰기 되게 된다. 선택 인자 length는 기본 문단 들여쓰기 값에 더해지는 값이다. altverse, patverse, patverse* 환경에서는 나누어진 행은 하나의 행으로 계산된다. \\> 매크로는 \verselinebreak를 간략히 쓴 것이다.

\begin{altverse} ... \end{altverse}

verse 환경 안에서 연들 사이는 입력시에 빈 줄로 구분한다. 그러나, verse 안에 놓인 각각의 연을 altverse 환경으로 둘러쌀 수도 있다. 그 결과는 \vgap 길이만큼 두번째, 네번째 행을 들여쓰기하는 것이다.

```
\begin{patverse} ... \end{patverse}
\begin{patverse*} ... \end{patverse*}
\indentpattern{⟨digits⟩}
```

altverse 환경과 달리, verse 환경 안에 놓인 각 연을 patverse로 둘러싸면, 각 행의 들여쓰기는 주어진 들여쓰기 패턴을 따른다. 이 패턴은 d_1에서 d_n까지의 한자릿수들로 구성된다. n번째 행은 $d_n\times$ vgap 만큼 들여쓰기된다. 첫번째 행은 d_1의 값과 상관없이 들여쓰지 않는다. 그리고 행의 숫자가 pattern 길이보다 크다면, 나머지 행들은 들여쓰기 되지 않는다.

patverse* 환경이 patverse와 다른 점은 들여쓰기 패턴이 환경의 끝까지 반복된다는 점이다.

patverse 환경의 들여쓰기 패턴은 \indentpattern 명령으로 지정한다. 여기서 ⟨*digits*⟩는 한자리수의 순차적 나열이다. 예를 들면 3213245281.

```
\linenumberfrequency{⟨nth⟩}
\thepoemline
\linenumberfont{⟨font-decl⟩}
\vrightskip
```

시의 각 행에는 번호를 붙일 수 있다. \linenumberfrequency 선언은 모든 ⟨*nth*⟩번째 행에 번호를 붙이도록 설정한다. ⟨*nth*⟩가 1보다 작으면 번호붙이기는 꺼진다. ⟨*nth*⟩가 1이면 모든 행에는 번호가 붙는다. 만약 ⟨*nth*⟩가 예컨대 5라면, 다섯번째 줄마다 번호가 붙는다. 기본값은 \linenumberfrequency{0}이다.

poemline 카운터가 행 번호에 사용된다. 행 번호의 식자 형식은 \thepoemline으로 지정하는데, 기본값은 아라비아 숫자이다. 이 숫자는 편집 영역의 끝에서 \vrightskip(기본값은 1em) 만큼 떨어져서 오른쪽 여백에 위치한다. 행 번호에 사용되는 글꼴은 \linenumberfont로 지정하는데, 기본값은 다음과 같다.

```
\linenumberfont{\small\rmfamily}
```

로만 폰트의 small 숫자로 식자되도록 하는 것이다.

```
\flagverse{⟨text⟩}
\vleftskip
```

\flagverse 매크로는 행의 행의 시작 부분에 사용할 수 있다. 행의 왼쪽 끝에서 \vleftskip (기본값은 3em)만큼 떨어져서 ⟨*text*⟩ 인자를 찍어준다. 이것은 예를 들면 연 번호를 붙이는 데 사용할 수 있다.

\poemtitle[⟨*short*⟩]{⟨*long*⟩}
\poemtitle*{⟨*long*⟩}
\poemtitlefont{⟨*font*⟩}

\poemtitle은 시 제목을 식자하고 ToC 엔트리로 추가한다. 별표붙은 \poemtitle*은 ToC 엔트리를 만들지 않는다. \poemtitlefont 매크로는 시 제목의 글꼴과 위치를 지정한다. 기본 정의는 다음과 같다.
\newcommand{\poemtitlefont}{\normalfont\bfseries\large\centering}
large bold centered 타이틀을 지정하는 것이다. 당연히 원한다면 바꿀 수 있다.

\beforepoemtitleskip
\afterpoemtitleskip

이 두 개의 길이값은 \poemtitle 앞뒤의 수직 간격을 나타낸다. 기본값은 \section 표제에서 사용되는 것과 동일한 간격으로 설정되어 있다. \setlength나 \addtolength로 바꿀 수 있다.

\poemtitlemark{⟨*title*⟩}

\poemtitle 매크로는 \poemtitlemark{⟨*title*⟩} 매크로를 부른다. \poemtitle*은 그렇게 하지 않는다. 기본값은 아무것도 하지 않는 것이다. 페이지 스타일 정의에서 (\sectionmark 또는 \chaptermark를 변경하는 것처럼) 바꿀 수 있을 것이다.

\poemtoc{⟨*sec*⟩}

\poemtitle 명령에 의해 ToC에 들어간 엔트리의 kind는 \poemtoc로 정의되어 있고, section과 같은 ToC 엔트리로 취급된다.
\newcommand{\poemtoc}{section}
이것은 예컨대 chapter나 subsection 등으로 바꿀 수 있다.

예제

이 클래스의 기능을 사용한 몇 가지 샘플을 보기로 하자.
먼저 옛날 식의 5행 속요(limerick)이다. 제목을 붙이고 중앙정렬했다.

```
\renewcommand{\poemtoc}{subsection}
\settocdepth{subsection}
\poemtitle{A Limerick}
```

```
\settowidth{\versewidth}{There was an old party of Lyme}
\begin{verse}[\versewidth]
There was an old party of Lyme \\
Who married three wives at one time. \\
\vin When asked: ‘Why the third?’ \\
\vin He replied: ‘One’s absurd, \\
And bigamy, sir, is a crime.’
\end{verse}
```

결과는 다음과 같다. \poemtoc를 재정의하여 subsection으로 설정하였으므로 \poemtitle 표제는 ToC에 번호없는 subsection으로 들어갈 것이다. 그러나, ToC는 section 이상만 열거하기 때문에 \settocdepth를 이용하여 여기서는 subsection 이상이 표제로 나오도록 하였다[2]

A Limerick

There was an old party of Lyme
Who married three wives at one time.
When asked: ‘Why the third?’
He replied: ‘One’s absurd,
And bigamy, sir, is a crime.’

다음은 altverse 환경으로 Garden 시를 만든다. 제목을 붙이고 중앙정렬하였다.

```
\settowidth{\versewidth}{But now my love is dead}
\poemtitle{Loves Lost}
\begin{verse}[\versewidth]
\begin{altverse}
\garden
\end{altverse}
\end{verse}
```

각각의 행이 어떻게 들여쓰기가 이루어지는지 다음 결과를 잘 보자.

Loves Lost

I used to love my garden
But now my love is dead

2 이 값은 이 장의 마지막에서 원래대로 되돌린다.

For I found a bachelor's button
In black-eyed Susan's bed.

시의 작자에 대한 정보를 추가하려 할 때 어떻게 할 것인가는 전적으로 사용자에게 달렸다. 여기 한 가지 예가 있다.

```
\newcommand{\attrib}[1]{%
   \nopagebreak{\raggedleft\footnotesize #1\par}}
```

이 명령은 다음 희시에서 사용되었다.

```
\poemtitle{Fleas}
\settowidth{\versewidth}{What a funny thing is a flea}
\begin{verse}[\versewidth]
What a funny thing is a flea. \\
You can't tell a he from a she. \\
But he can. And she can. \\
Whoopee!
\end{verse}
\attrib{Anonymous}
```

Fleas

What a funny thing is a flea.
You can't tell a he from a she.
But he can. And she can.
Whoopee!

Anonymous

다음 예는 긴 행이 어떻게 처리되는가를 시험한 것이다.

```
\poemtitle{In the Beginning}
\settowidth{\versewidth}{And objects at rest tended to remain at rest}
\begin{verse}[\versewidth]
Then God created Newton, \\
And objects at rest tended to remain at rest, \\
And objects in motion tended to remain in motion, \\
And energy was conserved
   and momentum was conserved
```

```
   and matter was conserved \\
And God saw that it was conservative.
\end{verse}
\attrib{Possibly from \textit{Analog}, circa 1950}
```

In the Beginning

Then God created Newton,
And objects at rest tended to remain at rest,
And objects in motion tended to remain in motion,
And energy was conserved and momentum was conserved and matter was conserved
And God saw that it was conservative.

Possibly from *Analog*, circa 1950

다음 예는 강제 행나눔을 시험한다. 타이틀 스타일도 약간 바꾸었다.

```
\renewcommand{\poemtitlefont}{\normalfont\large\itshape\centering}
\poemtitle{Mathematics}
\settowidth{\versewidth}{Than Tycho Brahe, or Erra Pater:}
\begin{verse}[\versewidth]
In mathematics he was greater \\
Than Tycho Brahe, or Erra Pater: \\
For he, by geometric scale, \\
Could take the size of pots of ale;\\ \settowidth{\versewidth}{Resolve by}
Resolve, by sines \\>[\versewidth] and tangents straight, \\
If bread or butter wanted weight; \\
And wisely tell what hour o' the day \\
The clock does strike, by Algebra.
\end{verse}
\attrib{Samuel Butler (1612--1680)}
```

Mathematics

In mathematics he was greater
Than Tycho Brahe, or Erra Pater:
For he, by geometric scale,
Could take the size of pots of ale;

Resolve, by sines
and tangents straight,
If bread or butter wanted weight;
And wisely tell what hour o' the day
The clock does strike, by Algebra.

Samuel Butler (1612–1680)

또다른 5행희시이다. 이번에는 patverse 환경의 장점을 살려본다. 여러 개의 5행 속요를 식자하는 경우 전체에 대하여 한 번만 \indentpattern을 설정하면 된다.

```
\settowidth{\versewidth}{There was a young lady of Ryde}
\indentpattern{00110}
\poemtitle{The Young Lady of Ryde}
\begin{verse}[\versewidth]
\begin{patverse}
There was a young lady of Ryde \\
Who ate some apples and died. \\
The apples fermented \\
Inside the lamented \\
And made cider inside her inside.
\end{patverse}
\end{verse}
```

The Young Lady of Ryde

There was a young lady of Ryde
Who ate some apples and died.
The apples fermented
Inside the lamented
And made cider inside her inside.

다음 예는 필자가 어릴 때부터 알고 있는 노래이다. 제 3 행에 나오는 'forty-niner'는 1849년의 골드 러시를 언급하고 있다.

```
\settowidth{\versewidth}{Oh my darling, Clementine}
\poemtitle{Clementine}
\begin{verse}[\versewidth]
\linenumberfrequency{3}
\flagverse{1.} In a cavern, in a canyon, \\
```

```
Excavating for a mine, \\
Lived a miner, forty-niner, \label{vs:1} \\
And his daughter, Clementine. \\!
\flagverse{\textsc{chorus}} Oh my darling, Oh my darling, \\
Oh my darling, Clementine. \\
Thou art lost and gone for ever. \\
Dreadful sorry, Clementine. \\!
\linenumberfrequency{0}
\end{verse}
```

Clementine

1. In a cavern, in a canyon,
   Excavating for a mine,
   Lived a miner, forty-niner,
   And his daughter, Clementine.

CHORUS Oh my darling, Oh my darling,
Oh my darling, Clementine.
Thou art lost and gone for ever.
Dreadful sorry, Clementine.

마지막 예는 `\indentpattern`을 과감하게 사용한 보기이다. 이 경우에는 다음과 같이 정의되어 있다.

```
\indentpattern{013555432211234689877977554565322234554445668877899}
```

결과는 다음 페이지를 보라.

Mouse's Tale

Fury said to
a mouse, That
he met
in the
house,
'Let us
both go
to law:
I will
prosecute
you. —
Come, I'll
take no
denial;
We must
have a
trial:
For
really
this
morning
I've
nothing
to do.'
Said the
mouse to
the cur,
Such a
trial,
dear sir,
With no
jury or
judge,
would be
wasting
our breath.'
'I'll be
judge,
I'll be
jury.'
Said
cunning
old Fury;
'I'll try
the whole
cause
and
condemn
you
to
death.'

Lewis Carrol, *Alice's Adventures in Wonderland*, 1865

십육 장

Verbatims, 박스, 파일

16.1 개요

이 장에서는 verbatim 텍스트, verbatim을 비롯한 여러 가지를 포함하는 새로운 박스들, 그리고 파일로부터 읽고 쓰는 것을 좀더 쉽게 하는 방법과 verbatim을 파일로 써서 번호를 붙여 읽어들이는 방법 등 새로운 기능에 대하여 논의한다.

16.2 Verbatims

shortvrb와 확장된 verbatim 패키지의 코드를 이 클래스에 포함하였다. 그리고 moreverb로부터도 일부를 빌어왔다.

verbatim 텍스트를 처리할 때 LaTeX은 문자에 어떤 특별한 의미가 있다고 생각하지 않는다. LaTeX은 주석 안에서 어떤 문자라도 무시한다.

`\newcomment{`⟨*comenv*⟩`}`

`\newcomment` 매크로는 새로운 comment 환경을 ⟨*comenv*⟩라는 이름으로 정의한다. ⟨*comenv*⟩ 환경 안에 오는 모든 텍스트는 무시된다.

`\begin{comment}` anything `\end{comment}`

이 클래스에서는 `comment` 환경이라는 한 가지 주석 환경을 제공하는데, 다음과 같이 정의되어 있다.

이것은 소스 문서의 제법 많은 부분을 '주석처리'하는 데 유용할 것이다.

\commentsoff{⟨*comenv*⟩} \commentson{⟨*comenv*⟩}

\commentsoff 선언은 ⟨*comenv*⟩ 주석 환경 안에서 코멘트 기능을 끈다. \commentson 선언은 ⟨*comenv*⟩ 주석 환경 안에서 코멘트 기능을 켠다. 각각의 경우 ⟨*comenv*⟩는 \newcomment 명령으로 미리 정의된 코멘트 환경이어야 한다.

예컨대, 다른 사람의 리뷰에 부칠 어떤 원고를 준비하고 있고, 리뷰할 사람을 위해서 약간의 주석을 부기하고 싶다고 하자. 또는 원고에 자신만이 알 수 있는 코멘트를 붙여두고 싶은 경우를 생각하자. comment 환경을 이용하여 개인적인 코멘트를 사용할 수 있고, 리뷰할 사람을 위해서는 다른 이름의 환경을 만들어둘 수도 있다. 이 주석들은 최종 문서에서는 나타나지 않도록 한다. 그러면 다음과 같이 소스를 작성할 수 있다.

```
\newcomment{review}
\ifdraftdoc\else
  \commentsoff{review}
\fi
...
\begin{comment}
Remember to finagle the wingle!
\end{comment}
...
\begin{review}
\textit{REVIEWERS: Please pay particular attention to this section.}
\end{review}
...
```

\verb 명령은 짧은 verbatim 텍스트를 표시하는 데 사용되는 명령이다. '짧다'고 하는 것은 한 줄을 넘어가지 않는다는 뜻이다. 매크로 명칭과 같은 작은 verbatim을 입력하려면 \verb?\amacro?와 같은 방법을 쓰게 된다. \verb* 매크로는 \verb와 비슷하지만 스페이스에 ␣ 표시를 넣는다는 점이 다르다.

\MakeShortVerb{⟨*verbchar*⟩} \DeleteShortVerb{⟨*verbchar*⟩}

\MakeShortVerb 선언은 \verb⟨*char*⟩와 같이 길게 쓰는 대신 한 문자만을 사용해서 verbatim을 표시할 수 있게 해준다. ⟨*verbchar*⟩ 인자는 백슬래시가 붙는 한 글자로 표시한다. \MakeShortVerb{\?}와 같이 선언하면 \verb?...?라고 쓰지 않고 ?...?로 써도 된다. ? 문자를 이와 같은 특별한 용도에 쓰지 않고 보통의 용법으로 되돌리려면, \DeleteShortVerb{\?} 한다.

\begin{verbatim} anything \end{verbatim} \begin{verbatim*} anything \end{verbatim*}

verbatim과 verbatim* 환경은 약간 긴 verbatim 텍스트를 조판하는 데 사용된다.

\setverbatimfont{⟨*font-decl*⟩}

모든 verbatim 텍스트에 사용될 폰트는 \setverbatimfont로 지정한다. 기본값은,
\setverbatimfont{\normalfont\ttfamily}
로 되어 있다. 만약 verbatim을 small 사이즈로 식자하고 싶다면 다음과 같이 하면 된다.

```
\setverbatimfont{\normalfont\small\ttfamily}
```

\tabson[⟨*num*⟩] \tabsoff

표준 verbatim 환경은 TAB을 무시하거나 여러 개의 TAB을 하나의 스페이스로 취급한다. 이 클래스에서는 \tabson 선언을 사용하면 TAB을 무시하지 않게 된다. \tabsoff는 verbatim 안의 tab 설정을 끈다. 기본값은 \tabsoff이다. 기본적으로 하나의 tab에 네 개의 스페이스를 할당하도록 되어 있는데, \tabson의 ⟨*num*⟩ 선택인자로 TAB에 해당하는 스페이스의 갯수를 지정할 수 있다.

\wrappingon \wrappingoff \verbatimindent \verbatimbreakchar

드문 일이기는 하지만, 긴 verbatim 라인을 다음 행으로 잘라서 넣고 싶을 때가 있다. \wrappingon 선언으로 verbatim 행이 마진까지 길어지면 다음 행으로 행나눔이 되도록 할 수 있다(\wrappingoff는 보통의 동작으로 되돌린다). 이렇게 나누어진 행의 두번째 이후 줄은 \verbatimindent 길이 만큼 들여쓰기되는데, 이 길이는 보통 방법으로 변경할 수 있다.

하나의 행이 다음 줄까지 이어지도록 하고 싶은 경우도 있을 수 있다. 이 역할은 \verbatimbreakchar로 지정하는데, 기본값은 다음과 같이 정의한 것이다.
\newcommand*{\verbatimbreakchar}{\char‘\%}
여기서는 나누어지는 행의 %를 두도록 하고 있다. 각 행의 끝에 /를 두게 하려면 다음과 같이 설정한다.

```
\renewcommand*{\verbatimbreakchar}{\char`\/}
```

tabbing과 wrapping 두 가지를 한꺼번에 사용하면 잘 되지 않는다는 점에 주의하라. 두 가지 중의 하나만 사용하는 것이 좋고 이왕이면 tabbing을 사용하는 것이 좋다.

verbatim에 대한 사항은 이 다음 절에서 조금 더 논의한다.

16.3 프레임 박스

박스에 프레임을 칠 수 있다. 이 안내문서의 구문 박스에서 그렇게 했다. 이것은 표준 LaTeX 클래스에도 있는 것이지만 거기에서는 페이지가 나누어지지 않았다.

`framed`, `shaded`, `leftbar` 환경은 페이지가 나누어진다. 이것은 framed 패키지 [Ars01b]에서 왔다.

```
\begin{framed} text \end{framed}
\begin{shaded} text \end{shaded}
\begin{leftbar} text \end{leftbar}
```

`framed` 환경은 텍스트를 `\fbox`와 비슷한 프레임 박스 안에 놓는다. 박스의 width는 text width와 같다. `shaded` 환경은 텍스트를 색깔이 칠해진 박스 안에 둔다. `leftbar` 환경은 텍스트의 왼쪽에 수직선을 그려준다. 모두 페이지 나누기가 이루어진다.

```
\FrameRule \FrameSep \FrameHeightAdjust
shadecolor
```

선의 두께는 `\FrameRule` 길이이고, 텍스트와 박스 사이의 간격은 `\FrameSep` 길이로 주어진다. 한 페이지 머리에서 베이스라인 위쪽 프레임 height는 `\FrameHeightAdjust` 매크로로 지정한다. 기본 정의는 아래와 같다.

```
\setlength{\FrameRule}{\fboxrule}
\setlength{\FrameSep}{3\fboxsep}
\newcommand{\FrameHeightAdjust}{0.6em}
```

`shaded` 환경의 배경색은 `shadecolor`로 지정하는데, 이것은 color 패키지 [Car98a]를 이용하여야 한다. 다음을 보라.

```
\usepackage{color}
\definecolor{shadecolor}{gray}{0.75}
```

`\frameasnormaltrue \frameasnormalfalse`

`\frameasnormaltrue` 선언에 의해 이 환경 안의 문단만들기는 메인 텍스트와 같이 설정될 것이다. `\frameasnormalfalse` 선언이 있으면 문단만들기가 `minipage`와 같아진다. 기본값은 `\frameasnormaltrue`이다.

프레임에 한 가지 알려진 문제가 있다. 프레임이 사용된 페이지의 페이지 헤더가 본문 활자보다 조금 작은 경우, 헤더가 약간 아래위로 이동하는 경우가 있다. 이것은 폰트 크기 설정을 그룹으로 묶어주면 피할 수 있다. 그러나 본문 폰트보다 큰 폰트를 사용할 때는 굳이 묶어줄 필요가 없다. 예를 들면 다음과 같다.

```
\makeoddhead{myheadings}{{\tiny Tiny header}}{}{\LARGE Large header}
```

이 환경은 좀 미묘해서, 플로트, 각주, 여백문단을 둘 수 없다. 그리고 표준 LaTeX이 제공하는 twocolumn 모드 이외의 다른 two column에서는 작동하지 않는다.

framed 패키지를 memoir 클래스와 함께 사용할 수도 있다. 이 경우에는 framed 패키지가 제공하는 기능이 클래스의 같은 기능을 하는 코드보다 우선적으로 적용될 것이다.

`\begin{fboxverbatim}` anything `\end{fboxverbatim}`

`fboxverbatim` 환경은 moreverb 패키지 [Fai98]가 제공하는 `boxedverbatim` 환경과 동일하고 이름만 다르다. 이 환경은 verbatim 텍스트를 식자하고 그 바깥쪽에 바짝 붙은 사각형 박스로 내용을 둘러싸준다.

`\begin{boxedverbatim}` anything `\end{boxedverbatim}`

`boxedverbatim` 환경은 `framed`와 `fboxverbatim` 환경을 한꺼번에 사용하는 것과 비슷하다. 그러나 그 나름의 기능을 조금 가지고 있다. 가장 단순하게 그냥 사용하면 내용을 verbatim으로 식자하고 `framed` 박스와 같은 사각형 박스(즉 페이지 나눔이 되는 박스)로 둘러싼다.

`\bvbox \bvsides \bvtopandtail \nobvbox`

`boxedverbatim` 환경에는 네 가지 프레임 스타일이 제공된다. `\bvbox` 선언은 기본 프레임 스타일을 사용하도록 하는 것인데, 사각형 박스를 그려준다. `\bvsides` 선언이 있으면 양쪽 변에만 선을 그리고 아래 위 선을 그리지 않는다. 반대로 `\bvtopandtail` 선언은 시작과 끝에만 가로선을 긋고 왼쪽 오른쪽 선을 그리지 않는다. `\nobvbox` 선언은 아무 선도 그리지 않게 한다.

\bvtopofpage{⟨*text*⟩}

기본 프레임 스타일(\bvbox)로는 한 페이지에 들어갈 내용이 너무 많아서 이어지는 박스의 상단에 헤딩을 둘 수 있다. 이 헤딩은 \bvtopofpage의 ⟨*text*⟩ 인자로 주어진다. 기본 설정은 아무것도 표시하지 않도록 \bvtopofpage{}로 하는 것이지만 원한다면 이전 페이지에서 이어져 온 verbatim 텍스트임을 다음과 같이 보여줄 수 있을 것이다.

```
\bvtopofpage{\begin{center}\normalfont (Continued)\end{center}}
```

\linenumberfrequency{⟨*nth*⟩} \linenumberfont{⟨*font-decl*⟩} \resetbvlinenumber

verse 환경의 경우와 똑같이, \linenumberfrequency의 ⟨*nth*⟩번째 인자 값이 0보다 크면, \linenumberfont를 이용해서 지정된 폰트로 행번호를 식자한다. 이 환경의 밖 어디에서든 행번호를 시작값으로 되돌리기 위해 \resetbvlinenumber를 지정할 수 있다.

\bvnumbersinside \bvnumbersoutside

행번호는 박스 안쪽(\bvnumbersinside)에 놓이거나 바깥쪽(\bvnumbersoutside)에 놓을 수 있다.

Verbatim tabbing은 boxedverbatim 환경에 적용된다. 그러나 wrapping은 적용되지 않는다.

16.4 파일

LaTeX에는 \input 명령이 있어서 파일을 읽어들일 수 있다. 그런데 어떤 내용을 파일에 쓰는 것에 대해서는 잘 알려져 있지 않다.

TeX은 16개의 '스트림'을 동시에 쓰기 위해 열 수 있다. 그리고 읽기 스트림도 16개가 있다. 이 스트림을 파일로 할당하면 데이터를 외부 파일로부터 읽거나 쓸 수 있다.

\newoutputstream{⟨*stream*⟩} \newinputstream{⟨*stream*⟩}

\newoutputstream 명령은 ⟨*stream*⟩이라는 이름을 갖는 새로운 쓰기 스트림을 만든다. 여기에는 텍스트와 명령이 전달될 수 있다. ⟨*stream*⟩ 인자는 반드시 공백없는 알파벳 문자로 이루어져야 한다. 마찬가지로, \newinputstream 명령은 파일로부터 읽기 위한 새로운 스트림을 만든다. ⟨*stream*⟩ 이름은 읽기 스트림과 쓰기 스트림이 서로 달라야 한다.

만약 너무 많은 스트림을 만들게 되면 TEX이 다음과 같은 에러 메시지를 보여줄 것이다.

\IfStreamOpen{⟨*stream*⟩}{⟨*true-code*⟩}{⟨*false-code*⟩}

만약 어떤 스트림 ⟨*stream*⟩이 열려 있는지를 확인하려면 \IfStreamOpen을 사용한다. 스트림이 열려 있으면 ⟨*true-code*⟩ 인자가 처리될 것이고, 그렇지 않고 스트림이 닫혀 있으면 ⟨*false-code*⟩ 인자가 실행된다.

\openoutputfile{⟨*filename*⟩}{⟨*stream*⟩}
\closeoutputstream{⟨*stream*⟩}

\openoutputfile 매크로는 ⟨*filename*⟩이라는 이름을 갖는 파일과 ⟨*stream*⟩ 쓰기 스트림을 연다. 그런 다음에 이 파일을 쓰기 스트림에 할당한다. ⟨*filename*⟩으로 이미 사용되던 내용이 있다면 모두 지워진다.

\closeoutputstream 매크로는 출력 스트림 ⟨*stream*⟩과 이 스트림에 할당된 모든 파일을 닫는다. 그런 다음 파일과 스트림의 연결을 끊는다.

\begin{writeverbatim}{⟨*stream*⟩} anything \end{writeverbatim}

writeverbatim 환경은 한 개의 인자를 취한다. 이 인자는 열려 있는 출력 스트림의 이름이다. 환경 안에 오는 내용은 이 스트림에 할당된 파일로 verbatim 출력되어 쓰여진다.

\addtostream{⟨*stream*⟩}{⟨*text*⟩}

\addtostream 명령은 ⟨*text*⟩를 이미 열려 있는 ⟨*stream*⟩ 출력스트림에 할당된 파일에 쓴다. ⟨*text*⟩에 포함된 명령은 쓰기 전에 모두 실행될 것이다. 이것을 방지하려면 명령이 확장되지 않도록 각 명령을 모두 \protect해주어야 한다.

\openinputfile{⟨*filename*⟩}{⟨*stream*⟩}
\closeinputstream{⟨*stream*⟩}

`\openinputfile` 매크로는 ⟨*filename*⟩이라는 이름을 갖는 파일과 ⟨*stream*⟩ 입력 스트림을 연다. 그리고 스트림을 파일에 읽기 할당한다. ⟨*filename*⟩이 존재하지 않으면 에러가 발생한다.

`\closeinputstream` 매크로는 입력 스트림 ⟨*stream*⟩과 거기에 할당된 파일을 닫는다. 그리고 파일과 스트림의 연결을 끊는다.

```
\readstream{⟨stream⟩}
\readaline{⟨stream⟩}
```

`\readstream` 매크로는 입력 스트림 ⟨*stream*⟩에 현재 할당된 파일의 내용을 읽어들인다. 이것은 `\input{`⟨*filename*⟩`}`이 하는 일과 동일하다.

`\readaline` 매크로는 TEX이 인식하는 방식의 한 줄을 현재 ⟨*stream*⟩ 입력 스트림에 할당된 파일로부터 읽어들인다. 몇 줄을 읽으려면 `\readaline`을 여러 번 사용하면 된다. 파일 끝에 도달해서 더이상 읽을 행이 없으면 경고(warning)를 낸다.

```
\verbatiminput{⟨filename⟩}
\readverbatim{⟨stream⟩}
\boxedverbatiminput{⟨filename⟩}
\readboxedverbatim{⟨stream⟩}
```

`\verbatiminput` 매크로는 `\input` 매크로와 비슷하지만 ⟨*filename*⟩ 파일의 내용을 verbatim text로 읽어오는 점이 다르다. 이와 유사하게 `\readverbatim`은 입력 스트림 ⟨*stream*⟩에 할당된 파일의 내용을 verbatim 텍스트로 읽어온다. 이 두 명령은 다음 명령과 구조적으로 유사하다.

```
\begin{verbatim}
  \input{...}
  \readstream{...}
\end{verbatim}
```

물론 다른 점이라면, 파일 내용이 실행되지 않는다는 점일 것이다.

`\boxedverbatiminput`은 이 모든 일을 종합한 것이다. ⟨*filename*⟩ 파일의 내용을 읽어들여서 테두리쳐진 verbatim 텍스트로 식자한다. 그리고 `\readboxedverbatim`은 ⟨*stream*⟩에 할당된 파일의 내용을 읽어들여서 박스쳐진 verbatim으로 식자한다.

tabbing, wrapping과 번호붙이기는 verbatim 환경에서 그러했듯이 verbatim input된 텍스트에도 적용할 수 있다.

Seventeen

Miscellaneous

여기서는 잡다한 많은 얘기를 하겠지만, ships, shoes, sealing wax, cabbages, kings에 대한 내용은 없다.

이 장은 *demo* 장 스타일과 *Ruled* 페이지 스타일을 사용하고, `\chapterprecis` 명령으로 시작한다.

17.1 들어가기

이 클래스는 몇 가지 추가 기능을 제공한다. 여기서는 그것에 대해 알아본다.

17.2 초고(draft documents)

draft 옵션을 주면 overfull이 발생한 행에 검은색 줄을 그어준다. 이 옵션이 없으면 final 문서가 된다.

`\ifdraftdoc`

`\ifdraftdoc` 매크로는 draft 문서를 처리할 때 추가로 지시하고 싶은 일을 설정할 수 있도록 해준다. 예를 들면 각 페이지 헤더(또는 바닥글)에 'DRAFT'라고 써두고 싶을 수도 있을 것이다. 다음 코드가 그런 일을 해준다.

```
\ifdraftdoc
  % special things for a draft document
\else
  % perhaps special things for a non-draft document
\fi
```

17.3 Change marks

원고를 준비할 때는 보통 여러 번에 걸쳐서 수정하는 일이 많다. 이 절에서 설명하는 매크로는 어떤 문서의 각 수정 단계에서 무엇이 달라졌는지를 확인하게 해주는 역할을 한다.

아래 구현된 명령은 iso 클래스 [Wil00b]의 마킹 변경 기능을 간략화한 것이다.

```
\changemarkstrue \changemarksfalse
```

change marking 매크로는 draft 클래스 옵션이 사용된 경우에만 작동한다. 그리고, 마크는 `\changemarkstrue` 선언이 효력이 있는 경우에만 인쇄된다. `\changemarksfalse`는 모든 마킹 스위치를 끈다.

```
\added{⟨change-id⟩}
\deleted{⟨change-id⟩}
\changed{⟨change-id⟩}
```

이 각각의 매크로는 마크를 표시하고 ⟨*change-id*⟩를 명령이 주어진 곳에서 가장 가까운 마진에 넣는다. `\added` 매크로는 원고에 무엇인가가 추가되었다는 것을 나타내고, $\oplus$를 기호로 사용한다. `\deleted` 매크로는 원고에서 어떤 내용을 제외하였음을 나타내고, $\neq$ 기호로 표시한다. `\changed` 매크로는 $\Leftrightarrow$ 기호로 표시되면 어떤 내용이 변했다는 것을 나타낸다.

마킹 명령들은 단어나 문장부호에 붙여서 공백없이 지시해야 한다. 그렇지 않으면 문서 전체에 추가 공백이 붙어나올 것이다.

17.4 Trim marks

memoir 클래스의 showtrims 옵션을 사용하면, 트림 마크가 각 페이지에 표시된다. 이것은 원래 의도했던 페이지 크기에 맞도록 종이를 잘라낼 자리를 표시하는 것이다.

트림 마크는 (트리밍된) 페이지의 귀퉁이와 양변의 가운데에 찍힌다.

```
\trimXmarks
\trimLmarks
\trimFrame
\trimNone
```

몇 가지 미리 정의된 트리밍 스타일이 있다. `\trimXmarks` 선언이 주어지면 페이지 네 귀퉁이의 십자표시가 표시된다. `\trimLmarks` 선언은 귀퉁이 표시의 모양을 'L'자 모양으로

바꾼다. 어느쪽 귀퉁이냐에 따라 이 모양의 방향이 변한다. \trimFrame은 모든 페이지의 페이지 경계에 해당하는 부분에 프레임을 그린다. \trimNone 선언은 트리밍 마크를 표시하지 않게 한다.

```
\trimmarks
\tmarktl \tmarktr \tmarkbr \tmarkbl
\tmarktm \tmarkmr \tmarkbm \tmarkml
```

\trimmarks 매크로는 모두 8개까지 트림 마크를 표시한다. 각 마크는 폭이 0인 그림으로 정의되어 있고 페이지의 경계를 잘 표시하도록 설계되어 있다.

\trimmarks 명령은 \tmarktl, \tmarktr, \tmarkbl, \tmarkbr을 각각 왼쪽상단, 오른쪽상단, 오른쪽하단, 왼쪽하단 귀퉁이에 찍어준다. \tmarktm, \tmarkmr, \tmarkbm, \tmarkml은 각각 상단중앙, 가운데 오른쪽, 하단중앙, 가운데 왼쪽에 온다. 이 \tmark.. 매크로는 폭이 0인 그림으로 확장된다.

```
\trimmark
```

\trimXmarks 선언은 \trimmark를 귀퉁이 십자표시로 사용한다. 이것은 다음과 같이 선언되어 있다.

```
\newcommand{\trimmark}{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1cm}\thicklines
    \put(-2,0){\line(1,0){4}}
    \put(0,-2){\line(0,1){4}}
  \end{picture}}
```

이 정의는 사이즈가 0인 4cm짜리 십자표시 그림이다.

예제로서, 페이지의 반 정도 길이에 해당하는 짧은 선을 그리고 싶다면, 다음과 같이 하면 된다.

```
\providecommand{\tmarkml}{}
\renewcommand{\tmarkml}{%
  \begin{picture}(0,0){%
    \unitlength 1mm
    \thinlines
    \put(-2,0){\line(-1,0){10}}
  \end{picture}}}
```

```
\providecommand{\tmarkmr}{}
\renewcommand{\tmarkmr}{%
  \begin{picture}(0,0){%
    \unitlength 1mm
    \thinlines
    \put(2,0){\line(1,0){10}}
  \end{picture}}}
```

10mm 길이의 얇은 수평선이 페이지의 경계 바깥쪽 2mm 위치에서 시작하여 가운데 왼쪽과 가운데 오른쪽에 표시될 것이다.

17.5 용지 번호

트림 마크의 한 가지 용도는 프린터에게 용지의 어디를 잘라야 하는지 보여주는 것이다. 또 한 가지 응용으로서, 각 페이지의 용지 번호를 표시하는 것도 유용하다. 용지번호란 첫 페이지를 1로 하여 페이지마다 1씩 증가시킨 것인데, 페이지 번호와는 별도로 매겨진다.

\thesheetsequence

\thesheetsequence 매크로는 현재 용지 번호를 식자한다. 이것은 \thepage 매크로의 사용법과 비슷하다.

lastsheet
lastpage

lastsheet 카운터는 지난번 LaTeX 실행시의 0이전 용지 숫자를 기억하고 있다. 마찬가지로 lastpage 카운터는 지난번 실행시의 마지막 페이지 번호를 기억하고 있다. 마지막 페이지의 번호가 마지막 용지 번호와 일치할 필요는 없다. 예를 들면,
이 문서에서, 이 페이지는 총 *369*장 중에서 *298*번째 용지인데, 전체 *339* 페이지 가운데 *268*번째 페이지이다.

앞의 문장은 다음 코드를 처리한 것이다.

```
\textit{이 문서에서, 이 페이지는 총 \thelastsheet 장 중에서
  \thesheetsequence 번째 용지인데, 전체 \thelastpage\ 페이지 가운데
  \thepage 번째 페이지이다.}
```

용지번호와 페이지번호를 트림 마크와 함께 표시하고 싶을 수도 있다. 다음과 같이 하면 페이지 상단에 용지번호가 표시된다.

```
\newcommand{\trimseqpage}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \put(0,2){\makebox(0,0)[b]{Sheet: \thesheetsequence\ of \thelastsheet}}
  \end{picture}}
\let\tmarktm\trimseqpage
```

17.6 리스트 앞에서 페이지 나누기

리스트(예를 들면 itemize)를 쓰기 전에 한두 개의 문장으로 리스트에 대해 설명하는 경우가 있는데, 이 때 이 앞에서 페이지 나눔이 이루어지면 이 도입부와 첫번째 아이템이 분리되어 불편하다.

```
\noprelistbreak
```

\noprelistbreak를 \begin{itemize} 직전에 두면 페이지 나누기를 억제한다. 리스트 시작 행과 그 앞의 도입부 사이에는 빈 줄이 없으면 좋다.

17.7 카운터 바꾸기

이 부분은 chngcntr 패키지 [Wil01e]를 이 클래스가 통합한 결과이다.

```
\newcounter{⟨ctr⟩}[⟨within⟩]
\thectr
```

LaTeX에서는 새로운 카운터, 예를 들어 ctr를 호출하려면 \newcounter{ctr} 형식으로 \newcounter 명령을 써서 정의한다. ⟨within⟩ 옵션인자가 주어지면 ⟨ctr⟩ 카운터는 within 카운터가 변경될 때마다 0으로 새로 설정된다. within 카운터는 미리 정의되어 있는 것이어야 한다. \thectr은 ctr 카운터의 값을 식자한다. 이 매크로는 \newcounter 명령이 사용될 때 자동으로 정의되어 있고 아라비아 숫자로 표시된다.

```
\counterwithin{⟨ctr⟩}{⟨within⟩}
\counterwithin*{⟨ctr⟩}{⟨within⟩}
```

\counterwithin 매크로는 \newcounter{ctr}로 정의된 ⟨ctr⟩ 카운터가 \newcounter{ctr}[within] 형식으로 정의된 것처럼 동작하게 만든다. \thectr 명령도 재정의해서 \thewithin.\arabic{ctr} 형식으로 변경한다. 별표붙은 명령은 \thectr을 변경하지 않고 원래 정의를 그대로 사용하게 한다.

\counterwithout{⟨*ctr*⟩}{⟨*within*⟩}
\counterwithout*{⟨*ctr*⟩}{⟨*within*⟩}

\counterwithout 매크로는 원래 \newcounter{ctr}[within] 형식으로 정의된 카운터가 \newcounter{ctr}로 정의된 것처럼 동작하게 만든다. \thectr 명령도 재정의하여 \arabic{ctr} 형식이 되게 한다. 별표붙은 명령은 \thectr을 재정의하지 않고 원래 정의를 그대로 사용하게 한다.

\counterwithin{ctr}{...}과 \counterwithout{ctr}{...} 명령은 여러 번 줄 수 있으므로, ctr을 두 가지 형식 사이에서 왔다갔다 사용할 수 있다. ctr의 현재 값은 이 명령에 영향을 받지 않는다. 현재 값을 바꾸려면 setcounter를 이용한다. 식자 형태를 바꾸려면 \thectr을 \renewcommand한다.

17.8 새로운 명령 정의하기

\newloglike{⟨*cmd*⟩}{⟨*string*⟩}
\newloglike*{⟨*cmd*⟩}{⟨*string*⟩}

이 클래스는 새로운 log-형식 수학 함수를 쉽게 만들 수 있다. 다음 예를 보라.

```
\newloglike{\Ei}{Ei}
```

별표붙은 명령은 limit 형태를 취하는 함수명(\max 함수 형식)으로 만들어준다.

LaTeX 커널은 \providecommand 매크로를 정의하고 있는데, 이것은 \newcommand와 같은 일을 하지만 이미 같은 이름의 명령이 정의되어 있을 때는 아무 것도 하지 않는다. 이 클래스는 여기에 몇 가지를 추가하였다.

\provideenvironment{⟨*name*⟩}[⟨*numargs*⟩][⟨*optarg*⟩]{⟨*begindef*⟩}{⟨*enddef*⟩}
\providelength{⟨*cmd*⟩}
\providecounter{⟨*ctr*⟩}[⟨*within*⟩]
\provideloglike{⟨*cmd*⟩}{⟨*string*⟩}
\provideloglike*{⟨*cmd*⟩}{⟨*string*⟩}

\provideenvironment, \providelength, \providecounter 매크로는 각각 \new... 명령에 상당한다. 같은 이름의 환경, 길이, 카운터가 정의되어 있지 않다면 새로 정의하고 그렇지 않으면 아무것도 하지 않으면서 경고 메시지를 낸다.

\provideloglike 명령은 수학의 log-형식 함수를 정의하는 것이다. 이 명령은 경고 메시지를 내지 않는다.

17.9 매크로 변경

간단한 매크로 정의를 확장하는 명령을 제공한다. patchcmd 패키지 [Dow00]를 이용하면 정의에 다른 것을 추가할 수 있다.

\addtodef{⟨*macro*⟩}{⟨*prepend*⟩}{⟨*append*⟩}
\addtoiargdef{⟨*macro*⟩}{⟨*prepend*⟩}{⟨*append*⟩}

\addtodef 매크로는 ⟨*prepend*⟩를 현재의 ⟨*macro*⟩ 정의 시작 부분에, 그리고 마지막에 ⟨*append*⟩를 추가한다. 여기서 ⟨*macro*⟩는 인자 없는 매크로 이름(백슬래시 포함)이다. \addtoiargdef 매크로는 이와 비슷하지만 한 개의 인자를 취하는 매크로 이름이다.

예를 들어서 다음 두 개의 \mymacro 정의를 각각 보자.

```
\newcommand{\mymacro}[1]{# is a violinist in spite of being tone deaf}


\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{}{ in spite of being tone deaf}
```

\addtoiargdef(\addtodef도 마찬가지) 명령은 동일한 ⟨*macro*⟩에 대해서 여러 번 적용할 수 있다. 앞의 예를 다시 사용하여 다음과 같이 할 수 있다.

```
\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{Although somewhat elderly, }%
                       { in spite of being tone deaf}
\addtoiargdef{\mymacro}{}{ and a bagpiper}
```

그 결과는 다음과 같이 정의한 것과 동일하다.

```
\newcommand{\mymacro}[1]{%
  Although somewhat elderly, #1 is a violinist
  in spite of being tone deaf and a bagpiper}
```

⟨*prepend*⟩와 ⟨*append*⟩ 인자는 LaTeX 코드를 포함할 수도 있다. 다음 예는 이 클래스에서 실제 사용된 예를 보인 것이다.

```
\addtoiargdef{\date}{}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\thanksmark}[1]{}
```

```
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\thedate{#1}
  \endgroup}
```

\addtoiargdef의 경우 인자에 대한 참조는 ⟨*macro*⟩에서의 경우와 동일하게 #1 등으로 할 수 있다.

\addtodef*{⟨*macro*⟩}{⟨*prepend*⟩}{⟨*append*⟩}
\addtoiargdef*{⟨*macro*⟩}{⟨*prepend*⟩}{⟨*append*⟩}

별표붙은 명령은 원래의 ⟨*macro*⟩가 \newcommand*로 정의되었을 경우 사용한다. 별표붙은 명령을 사용하는 것은 \renewcommand*와 같고 별표 붙지 않은 명령을 사용하는 것은 \renewcommand를 사용하는 것과 같다. 즉, \addto... 명령에 별표를 붙이거나 붙이지 않으면 각각 \renew...에 별표를 붙이거나 붙이지 않은 것과 동일하게 취급된다.

\addto... 매크로는 기존의 ⟨*macro*⟩에서 일부 코드를 삭제하는 데는 사용될 수 없다. 그리고 기존 매크로 정의의 처음과 끝이 아닌 위치에 어떤 것을 넣을 수도 없다. 그리고 일반적으로 옵션 인자를 취하는 매크로 정의나 별표붙은 매크로 정의를 바꾸는 데도 사용하지 못한다.

17.10 문자열 인자

이 클래스를 작성하면서 저자는 이따금 문자열로 이루어진 매크로 인자를 사용하였다. 예컨대 페이지 레이아웃 매크로(\settypeblocksize 등)의 * 인자나, \makeheadposition 매크로의 flushleft, center, flushright 인자 등이 그러하다.

\nametest{⟨*str1*⟩}{⟨*str2*⟩}
\ifsamename

\nametest 매크로는 ⟨*str1*⟩과 ⟨*str2*⟩의 두 개의 인자를 취한다. 만약 ⟨*str1*⟩이 ⟨*str2*⟩와 동일하면 \ifsamename을 TRUE로 하고 그렇지 않으면 FALSE로 한다. \nametest에서 비교하는 문자열은 공백을 포함할 수도 있고 \ 백슬래시 문자를 포함할 수도 있다. 두 개의 문자열이 완전히 일치하여야 동일한 것으로 판정한다.

저자는 이 매크로를 인자 값을 체크하기 위하여 주로 사용하였다. 다음은 보기이다.

```
\newcommand{\amacro}[1]{%
  \nametest{#1}{green}
  \ifsamename
```

```
%    code for green
  \fi
  \nametest{#1}{red}
  \ifsamename
%    code for red
  \fi
  ...
}
```

17.11 홀짝수쪽 페이지 검사

현재 페이지가 홀수인지 짝수인지를 엄격하게 검사하는 것은 쉽지 않은 일이다. 그러나 이 클래스는 페이지에 레이블을 붙이고 이 레이블의 참조를 체크하는 방법을 이용하여 엄격한 페이지 검사 방법을 제공한다. 이 검사가 성공하려면 적어도 LaTeX을 두 번 실행하여야 한다. 이 코드는 chngpage 패키지 [Wil01b]에서 가져온 것이다.

```
\checkoddpage
\ifoddpage
\strictpagechecktrue \strictpagecheckfalse
```

\checkoddpage는 페이지 번호가 홀수이면 \ifoddpage를 TRUE로 설정하고 짝수이면 FALSE로 설정한다. 엄격한 페이지 검사 방법은 레이블을 쓰고 읽는 코드를 포함하고 있으며, \strictpagechecktrue가 선언된 후에 활성화된다. 단순한 페이지 검사 방법은 TeX의 페이지 나누기 알고리듬의 특징 때문에 \checkoddpage 명령이 주어진 곳의 페이지 번호가 실제 페이지 번호와 일치하지 않을 수도 있다. 단순 검사법을 적용하려면 \strictpagecheckfalse를 선언하면 된다. 단순 검사가 속도는 더 빠르다.

```
\cplabel
```

엄격한 페이지 검사법이 사용되면 \cplabel로 정의된 것을 숫자 앞에 붙이는 레이블을 사용한다. 기본값은 ^_이다. 즉, 기본적으로 사용되는 레이블은 ^_21과 같은 문자를 포함하게 된다. 이 레이블이 다른 레이블과 충돌하는 경우에는 \cplabel을 \renewcommand 하면 되지만, 한 문서에서 이 정의를 중간에 바꾸지는 말아야 한다.

17.12 다른 페이지로의 이동

표준 LaTeX은 \newpage, \clearpage, \cleardoublepage 명령을 제공하는데, 모두 새로운 페이지를 시작하게 하는 명령들이다. 아래는 nextpage 패키지 [Wil00c]를 이 클래스가 포함한 결과이다.

\needspace{⟨*length*⟩}

이 매크로는 현재 페이지의 바닥에 ⟨*length*⟩ 공간이 있는지를 판단한다. 만약 공간이 있으면 아무것도 하지 않지만 그렇지 못하면 새로운 페이지를 시작한다. 이것은 ⟨*length*⟩ 만큼의 내용이 한 페이지에 함께 있도록 만들 때 유용하다. \needspace 매크로의 작동은 페널티에 의존하는데, 이것은 여분의 공간이 근사값이라는 의미이다. 그러나 특별한 경우를 제외하고는 만족할 만한 결과를 보여준다.

\Needspace{⟨*length*⟩}
\Needspace*{⟨*length*⟩}

\needspace와 비슷하게, \Needspace 매크로는 현재 페이지 바닥에 ⟨*length*⟩ 만큼의 공간이 있는지를 체크한다. 그리고 공간이 없으면 새로운 페이지를 시작한다. 이 명령은 문단과 문단 사이에서만 사용될 수 있다. 실제로 이 명령이 처음 하는 일은 \par를 부르는 것이다. \Needspace 명령은 페이지의 남은 공간을 계산할 때 실제 공간의 길이를 체크하므로 \needspace보다 더 정확하다.

\needspace나 \Needspace 명령이 새로운 페이지를 바로 만들게 되면 \flushbottom이 유효한 경우에도 ragged bottom으로 만들어진다. 별표붙은 명령 \Needspace*는 새로운 페이지가 \flushbottom이 유효하면 flush bottom으로, \raggedbottom이 유효하면 ragged bottom으로 만들어지도록 해준다.

일반적으로 말하자면 \needspace가 \Needspace보다 낫다. 다만 페이지 나눔이 아주 정확하지는 않고 페이지 flush bottom이 꼭 필요한 경우가 아니라면.

\movetoevenpage[⟨*text*⟩]
\cleartoevenpage[⟨*text*⟩]

\movetoevenpage는 현재 페이지를 중단하고 다음번 짝수쪽에서 타입세팅을 시작한다. \clear...는 다음번 짝수쪽으로 가기 전에 현재까지의 모든 플로트들을 모두 청산한다. 선택인자 ⟨*text*⟩는 중간에 건너뛰는 페이지가 하나 있게 되면 거기에 식자된다.

\movetooddpage[⟨*text*⟩]
\cleartooddpage[⟨*text*⟩]

이 매크로들은 \...evenpage 매크로들과 비슷하다. 다만, 다음번 홀수쪽으로 넘어간다는 점만 다르다.

마찬가지로, ⟨*text*⟩ 선택인자가 다음과 같이 주어지면,

```
\cleartooddpage[\vspace*{\vfill}THIS PAGE LEFT BLANK\vspace*{\vfill}]
```

새로운 홀수쪽이 시작되기 전에 한 페이지의 짝수쪽이 비워지게 될 때, 그 페이지 중앙에 'THIS PAGE LEFT BLANK'라는 문자열이 식자된다.

```
\cleartorecto \cleartoverso
```

이 명령은 \cleartooddpage와 \cleartoevenpage가 하는 일과 비슷하다[1]. 예컨대 Table of Contents가 *The TeXbook* [Knu84]에서와 같이 짝수쪽에 놓이게 하고 싶으면 다음과 같이 한다.

```
\cleartoverso
\tableofcontents
```

17.13 숫자 형식

숫자 표현 형식을 제어하는 방법이 몇 가지 제공된다. 두 가지 유형의 숫자 표현이 준비되어 있다. 'numeric number'는 아라비아 숫자로 표현하는 것이고 'named number'는 단어를 이용하는 방법이다.

숫자 형식 매크로의 인자는 '숫자'이다. 즉, 일련의 아라비아 숫자로 지시되는 것이어야 한다. 대표적인 인자 형식을 보면 다음과 같다.

- 숫자, e.g., \ordinal{123} -> 123rd
- 숫자로 확장될 수 있는 매크로, e.g., \def\temp{3}\ordinal{\temp} -> 3rd
- 카운터 값, e.g., \ordinal{\value{page}} -> 275th

  그러나, 카운터의 표현이 아라비아 숫자로만 이루어져 있지 않은, \thesection과 같은 경우에는 인자로 사용되면 다음과 같이 에러를 내거나 이상한 결과로 나타난다.

  \ordinal{\thesection} -> .1317th

1 조금 더 강력하다고나 할까.

Numeric 숫자

\cardinal{⟨*number*⟩}
\fcardinal{⟨*number*⟩}
\fnumbersep

\fcardinal 매크로는 ⟨*number*⟩ 인자를 \fnumbersep으로 세 자리마다 끊어서 표시해준다. \fnumbersep의 기본 정의는 다음과 같다.

```
\newcommand{\fnumbersep}{,}
```

예를 몇 가지 들어보자.

\fcardinal{12} -> 12

\fcardinal{1234} -> 1,234

\fcardinal{1234567} -> 1,234,567

\renewcommand{\fnumbersep}{ }\fcardinal{12345678} -> 12 345 678

\cardinal 매크로는 자릿수 사이를 분리하지 않는다는 점만 제외하고 \fcardinal과 같다.

\ordinal{⟨*number*⟩}
\fordinal{⟨*number*⟩}
\ordscript{⟨*chars*⟩}

\fordinal 매크로는 ⟨*number*⟩ 인자를 \fnumbersep에 의하여 자릿수 분리하고 서수형으로 표시한다. \ordinal 매크로는 자릿수 분리를 하지 않는다.

예를 들어본다.

\fordinal{3} -> 3rd

\fordinal{12341} -> 12,341st

\renewcommand{\ordscript}[1]{^{#1}}\fordinal{2} -> 2^{nd}

\ordinal{1234567} -> 1234567^{th}

This is the \ordinal{\value{chapter}} chapter. -> This is the 17^{th} chapter.

서수를 표시하는 글자는 \ordscript의 인자로 식자되는데, 기본값은 다음과 같이 정의되어 있다.

```
\newcommand{\ordscript}[1]{#1}
```

위에서 보인 대로, 이것은 다른 모양으로 바꿀 수도 있다.

```
\nthstring \iststring \iindstring \iiirdstring
```

서수 문자는 \nthstring 매크로의 값이다. 기본값은 th인데 대부분의 숫자에 적용된다. iststring은 기본값이 st로 되어 있고, 21st 등과 같이 1로 끝나는 서수에 적용된다. \iindstring은 기본값이 nd이고 22nd와 같이 적용된다. \iiirdstring은 기본값이 rd이고 23rd 등에 적용된다.

이름 숫자

```
\numtoname{⟨number⟩}
\numtoName{⟨number⟩}
\NumToName{⟨number⟩}
```

\numtoname 매크로는 ⟨*number*⟩ 인자를 소문자 단어로 식자한다. 나머지 두 개의 매크로는 유사한 것이지만 \numtoName은 첫 단어의 첫 글자를 대문자로 식자하고 \NumToName은 모든 단어의 첫 글자를 대문자로 식자한다.

예를 들어보자.

```
\numtoname{12345} ->
```
twelve thousand, three hundred and forty-five

```
\numtoName{12345} ->
```
Twelve thousand, three hundred and forty-five

```
\NumToName{12345} ->
```
Twelve Thousand, Three Hundred and Forty-Five

```
The minimum number in TeX is \numtoname{-2147483647}
    (i.e., \fcardinal{-2147483647}) ->
```
The minimum number in TeX is minus two billion, one hundred and forty-seven million, four hundred and eighty-three thousand, six hundred and forty-seven (i.e., -2,147,483,647)

```
\ordinaltoname{⟨number⟩}
\ordinaltoName{⟨number⟩}
\OrdinalToName{⟨number⟩}
```

이 세 개의 매크로는 \numtoname 등과 비슷하다. 다만 서수형 단어로 식자한다는 점이 다르다.

보기는 다음과 같다.

```
This is the \ordinaltoname{\value{chapter}} chapter. ->
```
This is the seventeenth chapter.

```
\namenumberand \namenumbercomma \tensunitsep
```

기본적으로 이름 숫자 표현에서 구둣점과 접속사가 사용된다. 미국 영어와 영국 영어 관행에 따르면, 하이픈이 십단위 이름(예, fifty)과 단단위 이름(예, two) 사이에 삽입된다. 이것은 \tensunitsep 값에 의하여 표현된다. 미국에서와 달리 영국에서는 콤마(\namenumbercomma)와 접속사(\namenumberand)를 필요한 위치에 삽입해야 한다. 이들 매크로는 다음과 같이 정의되어 있다.

```
\newcommand*{\namenumberand}{ and }
\newcommand*{\namenumbercomma}{, }
\newcommand*{\tensunitsep}{-}
```

다음 보기는 미국식 표현을 보여준다.

```
\renewcommand*{\namenumberand}{ }
\renewcommand*{\namenumbercomma}{ }
The maximum number in TeX is \numtoname{2147483647} (i.e., \cardinal{2147483647}). ->
```

The maximum number in TeX is two billion one hundred forty-seven million four hundred eighty-three thousand six hundred forty-seven (i.e., 2147483647).

\minusname \lcminusname \ucminusname

이름 숫자가 음수이면 \minusname이 숫자 표현 앞에 붙는다. 위의 세 명령의 정의는 다음과 같다.

```
\newcommand*{\lcminusname}{minus }
\newcommand*{\ucminusname}{Minus }
\let\minusname\lcminusname
```

이것은 'minus'를 일반적으로 소문자로 식자하는 것이다. 'minus'의 첫 글자를 대문자로 하려면, 간단히 다음과 같이 한다.

```
\let\minusname\ucminusname
```

\namenumberand에 별도의 명령이 제공되지 않는 것은 'and'가 'And'로 식자될 경우는 없을 것이라 보기 대문이다. 꼭 첫 글자를 대문자로 식자해야 한다면 매크로를 재정의하라.

이름 숫자를 표현하는 매크로는 많다. 영어 아닌 특정 언어에 적합한 이름 숫자를 식자하려면 이 매크로를 적절하게 수정해야 할 것이다. 클래스나 패치 코드에서 찾을 수

있을 것이다. 실제 이름의 선택과 순서는 내부 매크로 \n@me@number를 부름으로써 이루어진다. 만약 순서가 특정 언어에 적합하지 않다면 이 매크로를 수정해야 한다. 이 기능을 구현하는 데 더 효율적인 방법이 있다면 다음 번 릴리스 때 기본 정의를 변경할 것이다.

분수

텍스트에서 간단한 분수를 식자할 때는 두 가지 표현 방법이 가능하다. 하나는 3/4와 같이 하는 것이고, 다른 하나는 $\frac{3}{4}$처럼 하는 것이다. 어떤 것이 나으냐는 것은 맥락에 따라 달라진다. 이 분수 표시 방법은 다음과 같다.

```
... like 3/4 or $\frac{3}{4}$ ...
```

\slashfrac{⟨*top*⟩}{⟨*bottom*⟩}
\slashfracstyle{⟨*num*⟩}

이 클래스는 \slashfrac 명령을 제공하는데, 3/4와 같은 모양의 분수를 식자하게 해준다. 수학 모드에서만 사용할 수 있는 \frac 명령과는 달리 이 명령은 텍스트 모드와 수학 모드에서 모두 사용할 수 있다.

\slashfrac 매크로는 \slashfracstyle 명령을 호출하여 분수에 사용된 숫자의 크기를 줄인다. \slashfracstyle 명령 자체를 사용할 수도 있다.

```
In summary, fractions can be typeset like 3/4 or $\frac{3}{4}$
or \slashfrac{3}{4} or \slashfracstyle{3/4} because several choices
are provided.
```

요약하면 분수는 3/4, $\frac{3}{4}$, 3/4, 3/4 등으로 다양하게 표현할 수 있다.

^{⟨*super*⟩}
\textsubscript{⟨*sub*⟩}

숫자를 위나 아래로 이동시키기 위해서 커널 명령 \textsuperscript 매크로가 있다. 이것은 그 인자를 상첨자 위치에 식자한다. 이 클래스는 \textsubscript 매크로도 제공하는데, 인자를 하첨자 위치에 식자한다.

```
You can typeset superscripts like \textsuperscript{3}/4 and
subscripts like 3/\textsubscript{4},
or both like \textsuperscript{3}/\textsubscript{4}.
```

상첨자를 $^{3}/4$로, 하첨자를 $3/_{4}$로, 또는 둘 다 $^{3}/_{4}$로 식자하는 것이 가능하다.

17.14 array 데이터 구조

이 클래스에는 patverse 환경을 지원하는 몇 가지 매크로가 있는데, 이것을 일반적으로 사용할 수도 있다.

```
\newarray{⟨arrayname⟩}{⟨low⟩}{⟨high⟩}
```

\newarray는 ⟨*arrayname*⟩ 배열을 정의한다. 여기서 ⟨*arrayname*⟩은 MyArray와 같은 배열의 이름이다. 정수로 주어지는 ⟨*low*⟩와 ⟨*high*⟩ 인자는 각각 배열의 첫요소와 마지막 요소의 참조번호로 설정된다.

```
\setarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨text⟩}
\getarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨result⟩}
```

\setarrayelement 매크로는 ⟨*index*⟩의 ⟨*arrayname*⟩ 배열 내에서의 참조자에 해당하는 요소가 ⟨*text*⟩가 되도록 한다. 역으로, \getarrayelement는 ⟨*result*⟩라는 매크로에 ⟨*arrayname*⟩ 배열의 ⟨*index*⟩ 위치 요소의 내용을 저장한다.

```
\setarrayelement{MyArray}{23}{$2^{23}$}
\getarrayelement{MyArray}{23}{\result}
```

위의 예에서 \result는 \def\result{2^{23}}으로 정의한 것과 같다.

```
\checkarrayindex{⟨arrayname⟩}{⟨index⟩}
\ifbounderror
```

\checkarrayindex는 ⟨*arrayname*⟩이 배열이고 ⟨*index*⟩가 그 배열의 적절한 참조자인지를 검사한다. 두 조건이 모두 일치하면 \ifbounderror 값이 FALSE로 설정되고, ⟨*arrayname*⟩이 배열이 아니거나 ⟨*index*⟩가 범위를 넘어선 참조자이면, \ifbounderror는 TRUE로 설정된다.

```
\stringtoarray{⟨arrayname⟩}{⟨string⟩}
\arraytostring{⟨arrayname⟩}{⟨result⟩}
```

\stringtoarray 매크로는 ⟨*string*⟩의 각 문자들을 차례로 ⟨*arrayname*⟩ 배열에 참조자 1번부터 집어넣는다. \arraytostring 매크로는 ⟨*arrayname*⟩이 문자들로 이루어진 배열이라 간주하고 ⟨*result*⟩ 매크로를 그 문자들의 계열로 정의한다. 예를 들면 다음과 같다.

```
\stringtoarray{MyArray}{Chars}
\arraytostring{MyArray}{\MyString}
```

위의 예는 \def\MyString{Chars}와 동일하다.

\checkifinteger{⟨*num*⟩}
\ifinteger

\checkifinteger 명령은 ⟨*num*⟩이 0보다 작지 않은 정수인지를 검사한다. 검사가 성공하면 \ifinteger는 TRUE가 되고, 실패하면 FALSE가 된다.

17.15 pdfLaTeX

동일한 문서를 LaTeX과 pdfLaTeX으로 동시에 실행해야 할 때가 있다. LaTeX은 .dvi 파일을 출력하지만 pdfLaTeX은 .dvi나 .pdf 파일을 출력한다.

pdfLaTeX의 기본값은 .dvi 파일을 출력하는 것이지만 시스템의 설정 파일에 의하여 pdfLaTeX의 동작이 .pdf 출력으로 맞추어져 있을 수 있다. .pdf 출력을 강제하려면 preamble 시작 부분에 \pdfoutput=1 이라는 문장을 써주어야 한다. 반대로 .dvi 출력을 강제하려면 \pdfoutput=0 으로 설정해둔다. 이 명령은 LaTeX이 알지 못하는 것으로 LaTeX을 실행하면 불평할 것이다.

\ifpdf ... \fi

이 클래스는 \ifpdf 조건문을 제공한다. 이것이 참이면 문서는 pdfLaTeX에 의하여 처리되는 것이고 그렇지 않으면 거짓이 된다. 다음과 같이 사용할 수 있다.

```
\ifpdf
  code for pdfLaTeX only e.g., \pdfoutput=1 or \pdfoutput=0
\else
  code for LaTeX only
\fi
```

LaTeX에만 적용할 코드가 없다면 \else 부분은 쓰지 않아도 된다.

17.16 행간

LaTeX은 폰트 사이즈가 달라지면 서로 다른 행간을 적용한다.

```
\baselineskip \onelineskip
```

문서의 어디에서든지 표준 LaTeX길이 \baselineskip에 현재의 기본행간 값이 저장되어 있다[2] 이 클래스는 \onelineskip 길이를 제공하는데, 여기에는 normal font에 해당하는 기본 행간 값이 저장되어 있다. 이 값은 특정한 길이값을 텍스트의 행수로 지정하고자 할 때 유용하다.

17.17 간격 조절

커널 명령 \,는 텍스트 모드와 수학 모드에서 thin space를 넣는 데 쓰인다. 그밖의 간격 조절 명령으로 음수값 thin space인 \!와 medium space \:, thick space \;들이 있는데 수학 모드에서만 사용할 수 있는 것들이다.

```
\thinspace \medspace \: \!
```

우연히, 수학 모드에서처럼 텍스트에서도 간격을 주는 명령을 사용하면 좋겠다는 생각이 들었다. 커널 매크로 \thinspace는 3/18 em에 해당하는 thin space를 나타낸다. 이 클래스에서는 \medspace라는 4/18 em의 medium space 명령을 제공한다. 앞서 말했듯이 커널 매크로 \:는 수학모드에서 medium space를 넣는 데 쓰인다. 이 클래스에서는 수학모드와 텍스트 모드 모두 medium space를 넣을 수 있도록 했다. 그리고 이 클래스의 \!, 음수값 thin space도 텍스트와 수학식에서 모두 사용할 수 있도록 했다.

math thick space는 5/18 em이다. 텍스트 모드에서 이 값을 얻으려면 간격 명령을 결합해서 사용해야 한다.

```
\:\:\!
```

위와 같이 하면 5/18 em의 간격을 얻을 수 있다$((4+4-3)/18)$.

17.18 상호참조

LaTeX의 상호참조 명령은 \ref와 \pageref이다. 각각 label을 참조하거나 그 label이 있는 페이지를 참조한다.

```
\fref{⟨label⟩} \figurerefname
\tref{⟨label⟩} \tablerefname
\pref{⟨label⟩} \pagerefname
```

2 baseline을 stretch하는 경우를 무시하고 하는 말이다.

이 클래스는 그림이나 표, 페이지와 같은 이름을 붙인 특별한 참조 명령을 제공한다. 예를 들어 \fref와 \pref의 기본 정의는 다음과 같다.

```
\newcommand{\fref}[1]{\figurerefname~\ref{#1}}
\newcommand{\pref}[1]{\pagerefname~\pageref{#1}}
```

이 명령은 다음과 같이 사용하고,

```
\ldots footnote parameters are shown in~\fref{fig:fn} on~\pref{fig:fn}.
```

결과는 다음과 같다.

> ...footnote parameters are shown in 그림 (13.1) on 216 쪽.

\Pref{⟨*label*⟩} \partrefname
\Cref{⟨*label*⟩} \chapterrefname
\Sref{⟨*label*⟩} \sectionrefname

Part(\Pref), Chapter(\Cref), section(\Sref)에 대한 이름 붙은 참조 명령도 제공된다.

```
\newcommand{\Sref}[1]{\sectionrefname\ref{#1}}
```

이 문서에서 다음과 같이 쓰였다.

```
‘In \Cref{chap:misc} there is a section
(\Sref{sec:xrefthis}) about cross references.’
```

결과는 다음과 같다.

> ‘In 제 17 장 there is a section (§17.18) about cross references.’

【memhangul】 한글 문서에서는 이 상호참조명령이 조금 바뀌어야 한다. memhangul-ucs는 이 명령들을 재정의해두고 있다. ■

문서의 일부를 번호가 아닌 이름으로 참조하는 것이 유용할 때가 있다. 예컨대

```
The chapter \textit{\titleref{chap:bringhurst}} describes \ldots
```

The chapter *An example design* describes ...

번호가 아닌 이름 참조를 가능하게 하는 패키지는 두 개가 있다. nameref [Rahtz01]과 titleref [Ars01a]이다.

이름 참조는 장절 명령의 두번째 옵션 인자를 추가한 결과 이 클래스에 추가된 기능이다. 이것은 nameref 패키지와 따라서 hyperref 패키지와도 충돌하였기 때문에 그것이 수정되어야 했다. 수정결과 Donald Arseneau의 titleref 패키지와도 충돌하게 되었고 nameref은 titleref과 맞지 않았다. 이 클래스는 이들 패키지가 알지 못하는 \poemtitle과 같은 타이틀도 제공하기 때문이다. 그래서 차라리 이름 참조 그 자체를 클래스에서 구현하는 것이 더 낫겠다고 판단하게 되었다.

`\label{`⟨*key*⟩`}` `\ref{`⟨*key*⟩`}` `\pageref{`⟨*key*⟩`}`
`\titleref{`⟨*key*⟩`}`
`\headnamereftrue` `\headnamereffalse`

\titleref 매크로가 일반적인 상호참조 명령에 추가되었다. 이것은 번호가 아니라 label 번호와 관련된 title을 식자한다. 물론 이것은 관련된 타이틀이 있는 경우, 즉 \caption, \section 등의 경우에만 사용가능하다. 좋지 않은 보기를 들자면,

```
Labelling for \verb?\titleref? may be applied to:
\begin{enumerate}
\item Chapters, sections, etc.       \label{sec:xref:item1}
...
\item Items in numbered lists, etc. \ldots \label{sec:xref:item3}
\end{enumerate}
Item \ref{sec:xref:item2} in section~\ref{sec:xref} mentions captions
while item \titleref{sec:xref:item3} in the same section
\textit{\titleref{sec:xref}} lists other things.
```

위의 코드가 만드는 결과는 다음과 같다.

Labelling for \titleref may be applied to:

1. Chapters, sections, etc.
2. Captions
3. Legends
4. Poem titles
5. Items in numbered lists, etc.

Item 2 in section 17.18 mentions captions while item 상호참조 in the same section 상호참조 lists other things.

이 예에서 보듯이, \titleref를 사용할 때는 주의할 점이 있다. 일반적으로 말하자면 \titleref{⟨*key*⟩}는 ⟨*key*⟩를 정의하는 \label 이전의 마지막 이름을 찍는다.

chapter와 하위수준 장절명령들은 세 개의 서로 다른 title text를 가지고 있다. main title, ToC용 title, 페이지 헤더용 title이 그것인데, 기본값인 \headnamereffalse에서는 ToC 타이틀이 \titleref에서 식자된다. \headnamereftrue를 선언하면 페이지 헤더용 텍스트가 사용된다.

NOTE: 특히, memoir 클래스를 사용할 때는 \label 명령을 \chapter, \section, \part 등 명령의 인자 안에 두어서는 안된다. 이렇게 하면 그것은 무시되거나 잘못된 값으로 식자될 것이다. 표준 클래스에서는 이런 제한이 없지만, \label 명령을 인자로 포함시키는 것은 그다지 좋은 관행은 아니라고 생각한다.

\currenttitle

현재의 타이틀을 참조하고 싶다면 \currenttitle을 쓸 수 있다. 이 명령은 타이틀과 결부된 label이 이미 있고 \titleref이 그 타이틀을 참조하는 것과 마찬가지로 동작한다. 예를 들면 다음과 같다.

```
This sentence in the section titled ‘\currenttitle’ is an example of
the use of the command \verb?\currenttitle?.
```

This sentence in the section titled ‘상호참조’ is an example of the use of the command \currenttitle.

\theTitleReference{⟨*num*⟩}{⟨*text*⟩}

\titleref과 \currenttitle은 모두 \theTitleReference를 이용하여 타이틀을 식자한다. 이것은 두 개의 인자로 불리는데, 하나는 ⟨*num*⟩ 번호이고 다른 하나는 타이틀 텍스트인 ⟨*text*⟩이다. 기본 정의는 다음과 같이 되어 있다.

```
\newcommand{\theTitleReference}[2]{#2}
```

이 정의에 따라 ⟨*text*⟩ 인자가 인쇄된다. 이것은 예컨대 다음과 같이 바꿀 수 있다.

```
\renewcommand{\theTitleReference}[2]{#1\space \textit{#2}}
```

이렇게 하면 숫자가 먼저 나오고 그 다음에 이탤릭체로 타이틀이 식자된다. 이런 식으로 할 때는 \titleref를 번호붙은 타이틀에만 적용해야 하는데, 번호 없는 타이틀에서 번호를 식자하는 것은 (a) 무의미한 일이고, (b) 올바른 결과를 얻을 수도 없기 때문이다.

\titleref, \theTitleReference, \currenttitle 명령들은 titleref 패키지 [Ars01a]의 해당 명령에 직접 대응한다.

\namerefon \namerefoff

이름 참조로 인해 몇 개의 인자가 착종을 일으키는 부작용이 발생한다. \legend 명령의 경우가 그렇다. 이름 참조를 꼭 써야할 상황이 아니라면 \namerefoff 선언으로 이것을 꺼두도록 하자. \namerefon 선언은 이름 참조를 다시 활성화한다.

17.19 단어와 구

이 클래스에는 단어나 구절로 치환되는 매크로가 몇 개 있다. 다른 언어로 식자할 때는 이것을 변경해야 할 것이고, 편집자나 저자에 따라서 영어 표현이라도 다른 것으로 바꾸고자 할 때가 있을 것이다. 다음 리스트는 이 매크로의 기본값을 보여준다.

\contentsname 차 례
\listfigurename 그림 차례
\listtablename 표 차례
\abstractname 요 약
\partname 부
\chaptername 장
\appendixname 부록
\appendixtocname 부 록
\appendixpagename 부 록
\bibname 참고 문헌
\indexname 찾아보기
\figurename 그림
\tablename 표
\figurerefname 그림
\tablerefname 표
\pagename 페이지
\pagerefname 페이지
\partrefname 편(defined as \newcommand{\partrefname}{Part~})
\chapterrefname 장(defined as \newcommand{\chapterrefname}{Chapter~})

`\sectionrefname` §(defined as `\newcommand{\sectionrefname}{\S}`)

위의 정의는 매우 단순하다.

```
\newcommand{\partname}{Part}
```

그리고 바꾸는 것도 간단할 것이다.

이름과 숫자에 대한 매크로 정의는 조금 더 복잡하다. 예를 들어 11(eleven)에 해당하는 정의는 다음과 같다.

```
\newcommand*{\nNamexi}{\iflowertonumname e\else E\fi leven}
```

즉, 각각의 정의는 첫 글자가 대문자인지 소문자인지를 판별하기 위해 조금 복잡한 방법을 쓰고 있는 것이다. 더 자세한 것은 클래스의 코드에 관한 문서를 읽어보도록 하라.

17.20 기호문자

LaTeX은 매우 다양한 기호문자를 식자할 수 있게 해준다. 이 클래스는 표준 LaTeX에 이 점에 대해서는 새로운 것을 추가하지 않았다. 어떤 기호들을 사용할 수 있는지에 대해서는 Scott Pakin의 *The Comprehensive LaTeX Symbol List* [Pak01] 을 읽어보면 된다. 원하는 기호를 찾는 데 약간의 노력이 들지도 모르기는 하지만.

예를 들어 `\texttrademark` 명령은 trademark 기호 ™를 식자한다. 그러나 `\textregistered` 명령은 ®를 식자한다. registered trademark를 식자하려 할 때는 이것이 베이스라인이 아니라 상첨자 위치에 오기를 원하는데, 이럴 경우 `\textsuperscript` 매크로를 사용해야 한다.

```
\textsuperscript{\textregistered}
```

이렇게 하면 ®를 얻을 수 있다.

18.1 INTRODUCTION

이 장에서는 완전한 디자인 과정을 수행해보려 한다. 이 장 자체가 이 작업의 결과 만들어진 형식으로 조판되었다.

무슨 새로운 것을 창안해내기보다 Bringhurst의 책 *The Elements of Typographic Style* [Bri92]에 적용된 디자인을 토대로 작업하려 하는데, 이 책의 디자인은 일반적인 클래스를 사용하여 만든 LaTeX 문서의 모양과는 전혀 다르고, 또한 훌륭한 타이포그라피를 주창하는 저자가 만든 디자인이기도 하기 때문이다.

가능한 한, 이 장의 조판은 코딩과 그림 예제를 곁들여서 결과를 제시하려 한다.

18.2 DESIGN REQUIREMENTS(디자인 기본요소)

*Elements of Typographic Style*은 본문에 Minion 폰트를, 캡션에는 Syntax (산세리프 폰트 가운데 하나) 폰트를 쓰고 있다. 페이지 레이아웃은 19 쪽의 그림 (2.2)에 보인 바와 같다. 그러나 똑같이 만들려면 좀더 자세한 기술이 필요하다.

페이지의 잘린 크기는 23 × 13.3cm이다. foredge는 3.1cmd이고 top margin은 1.9cm 이다.

앞서 말한 대로 본문 폰트는 Minion인데, 21pc 문단폭, 12pt 행간에 페이지당 42행이 들어 있다. 우리는 Minion이 10pt Computer Modern Roman (이 안내 문서의 기본 폰트) 폰트로 대체될 수 있는 것으로 가정하겠다. 그림과 표의 캡션은 캡션이름과 번호를 붙이지 않고 Syntax 폰트로 식자되고 있다. 캡션은 보통 그러하듯이 본문보다 조금 작은 폰트로 식자되었는데, 우리는 캡션 폰트에 `\small \sfseries` CMR 폰트로 식자하는 것으로 한다.

바닥글(footer)은 편집영역과 같은 폭이고 면주(folio)는 바닥글의 foredge에 위치한다. 편집영역의 아래쪽 끝과 folio 사이에는 2행간의 간격이 있다.

글자 그대로의 의미에서 헤더는 없다. 그러나 장 타이틀은 홀수쪽 페이지에 놓이고 절 타이틀은 짝수쪽 페이지에 놓인다. 이 페이지마다 표시되는 타이틀은 편집 영역 일곱번째 줄 위치에 바깥쪽 마진에 놓인다. 홀수쪽 헤더는 raggedright 정렬되고 짝수쪽 헤더는 raggedleft된다.

Bringhurst는 많은 마진 주석을 사용하는데, 가장 긴 폭은 51pt이다. 조금 작은 글씨체로 raggedright 정렬되어 식자된다.

장 타이틀은 small caps 소문자로, 본문보다는 큰 폰트를 사용한다. 그리고 타이틀과 편집영역 사이에 줄을 긋는다. 장 타이틀이 차지하는 수직 공간은 텍스트 세 줄에 해당한다. 본문의 장 타이틀에는 번호가 붙지 않는데 목차에는 붙는다.

절 타이틀은 역시 small caps 소문자인데 본문 폰트와 같은 크기이다. 타이틀에는 장과 절의 번호를 모두 표시하는 번호가 붙는다.

subsection 타이틀은 이 책의 장절구분에서 가장 하위단위이다. 본문 글꼴을 이탤릭 처리하고 번호붙은 들여쓰기없는 문단으로 식자된다. Bringhurst는 이것을 가끔 enumerated list처럼 사용하기 때문에 이어지는 텍스트 없이 subsection 타이틀만 여러 줄로 나올 때도 있다.

장 타이틀만이 ToC에 들어간다. 그리고 타이틀 바로 뒤에 페이지 번호가 붙어 나오고 raggedright 식자된다. LoF나 LoT는 없다.

일반적인 LaTeX 폰트 사용과는 달리, 단지 세 가지 크기의 폰트만이 사용된다는 데 주

의하라. 즉, 텍스트 폰트, 그보다 조금 큰 chapter 타이틀, 그리고 그보다 조금 작은 마진 노트와 캡션에 쓰인 폰트. 그리고, 두꺼운 글꼴은 특별한 경우 외에는 쓰이지 않았다. 특별한 경우란, 글꼴 가족을 비교할 때와 차이점을 쉽게 알아보게 하기 위해 큰 볼드 글꼴을 쓸 때이다.

18.3 SPECIFYING THE PAGE AND TYPEBLOCK

Specifying the page and typeblock

첫번째와 그 다음으로 해야 할 일은 용지를 잘라내어 페이지 크기를 정하고 편집 영역을 설정하는 것이다. 트리밍된 크기는 쉽게 그 크기를 지정할 수 있다.

```
\settrimmedsize{23cm}{13.3cm}{*}
```

그런데 편집 영역의 길이(height)를 행수로 설정하려면 약간의 트릭을 써야 한다. 높이 계산을 하여 편집 영역에 정수배의 행수가 들어가도록 해야 하는 것이다. 그런 다음에 마지막 줄의 행높이를 고려하여 약간의 높이를 추가해준 다음 마지막 줄의 베이스라인에서 첫째줄의 상단까지 길이값을 계산해야 한다. 만약 N행을 넣으려면 $N-1$배만을 계산해야 하는 것인데, 우리는 42행이 들어가게 할 생각이므로 편집영역의 높이가 기본 행간(즉 `\onelineskip`)의 41배가 되도록 하면 된다.

```
\settypeblocksize{41\onelineskip}{21pc}{*}
```

좀 쉽게 하기 위해서 용지의 상단을 트리밍하지 않기로 한다.

```
\setlength{\trimtop}{0pt}
```

그러나 바깥쪽 여백은 잘라내어야 한다. 먼저 `\trimedge`에 `\stockwidth` 값을 할당한 다음 여기에서 `\paperwidth`를 빼면 `\trimedge` 길이가 잘라낼 foredge 크기가 된다.

```
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

트리밍된 페이지와 편집영역의 크기가 정해졌다. 편집영역도 페이지 상에 설정되었다. foredge margin이 3.1cm가 되게 하면 좌우여백 크기도 정해진다.

```
\setlrmargins{*}{3.1cm}{*}
```

위쪽 마진은 1.9cm로 설정한다. 이것은 네 줄 반 정도에 해당한다. 폰트 사이즈를 다르게 하고자 하는 분들을 위해서, 나는 위쪽 마진을 폰트 사이즈에 의존하도록 하려 한다. `\footskip`도 여기에서 설정할 수 있다. (이것은 헤더에 관련된 길이가 아니고 편집영역에도 상관없다.)

```
\setulmargins{4.5\onelineskip}{*}{*}
\setheadfoot{\onelineskip}{3\onelineskip}
\setheaderspaces{\onelineskip}{*}{*}
```

마지막으로 마진 노트를 위한 길이들을 설정한다.

```
\setmarginnotes{17pt}{51pt}{\onelineskip}
```

이것으로 되었다고 생각되면 페이지 레이아웃을 체크하고 적용하게 한다.

```
\checkandfixthelayout
```

이 레이아웃을 이 chapter에 적용하는 것은 불가능하지 않지만, 여기서 시도할 생각은 없다. 그리고 어떻게 할 수 있는가에 대해서는 말하지 않겠다. 아주 특별한 경우가 아니면 한 문서의 중간에서 페이지 레이아웃을 이렇게 근본적으로 바꾸어버리는 것은 결코 좋은 생각이 아니다. 그러나, 292 쪽에 있는 그림은 이러한 레이아웃이 US letterpaper 용지상에서 어떻게 보이는가를 보여주고 있다. 이 그림을 잘 보면 용지가 페이지 크기로 trim down되지 않는다면 이상하게 보일 수 있음을 알 수 있을 것이다. 여기에서 바로 레이아웃을 바꾸지 않은 또하나의 이유이다.

An example design

18.4 SPECIFYING THE SECTIONAL TITLING STYLES

The chapter style

다시 정리하면, chapter 타이틀은 소문자의 small caps로 본문보다 큰 폰트를 사용한다. 타이틀과 편집영역의 사이에는 횡선이 들어간다. chapter title이 차지하는 공간은 세 줄 분량이다. 본문 중의 chapter에는 번호가 붙지 않지만 목차에는 붙어 있다. ToC의 타이틀은 대소문자를 모두 사용한다.

chapter style을 정의하는 것은 아주 쉽다. 다음을 보라.

```
%% Bringhurst chapter style
\makechapterstyle{bringhurst}{%
  \renewcommand{\chapterheadstart}{}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\printchapternum}{}
  \renewcommand{\afterchapternum}{}
  \renewcommand{\printchaptertitle}[1]{%
    \raggedright\Large\scshape\MakeLowercase{##1}}
  \renewcommand{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip\onelineskip}
}
```

대부분의 LaTeX 설정들을 비우고 단지 두 개만 수정하면 된다.

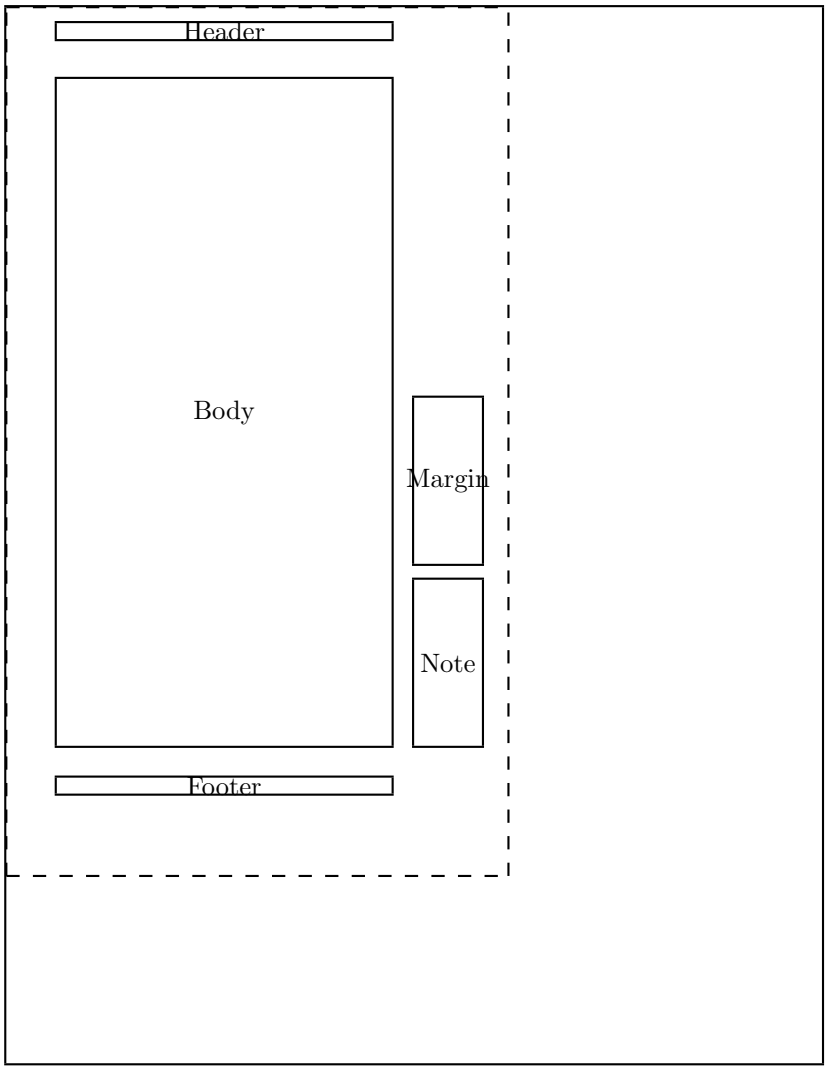
chapter title(`\printchaptertitle`)은 `\Large` smallcaps 폰트로 raggedright 식자한다. `\MakeLowercase` 매크로는 전체 타이틀이 식자되기 전에 소문자로 바뀌도록 해준다. 실제 타이틀의 입력은 대소문자를 섞어서 한다.

타이틀을 식자한 후에 `\afterchaptertitle` 매크로로 한 행 길이를 스킵하고 횡선을 그은 다음에 다음 한 줄을 스킵한다.

【memhangul】 memhangul-ucs에서는 다음과 같이 하여야 한다.

```
%% Bringhurst chapter style
\makechapterstyle{bringhurst}{%
  \renewcommand{\chapterheadstart}{}
```

Dashed lines represent the actual page size after trimming the stock.



Lengths are to the nearest pt.

`\stockheight = 795pt`	`\stockwidth = 614pt`
`\pageheight = 654pt`	`\pagewidth = 378pt`
`\textheight = 502pt`	`\textwidth = 252pt`
`\trimtop = 0pt`	`\trimedge = 236pt`
`\uppermargin = 54pt`	`\spinemargin = 38pt`
`\headheight = 12pt`	`\headsep = 30pt`
`\footskip = 36pt`	`\marginparsep = 17pt`
`\marginparpush = 12pt`	`\columnsep = 10pt`
`\columnseprule = 0.0pt`	

An illustration of Bringhurst's page layout style when printed on US letter paper stock. Also shown are the values used for the page layout parameters for this design.

```
%% \printchaptername is not used in memhangul-ucs
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\printchapternum}{}
  \renewcommand{\afterchapternum}{}
%% next two lines added., tr.
  \renewcommand{\prechapternum}{}
  \renewcommand{\postchapternum}{}
  \renewcommand{\printchaptertitle}[1]{%
    \raggedright\Large\scshape\MakeLowercase{##1}}
  \renewcommand{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip\onelineskip}
}
```

An example design

즉, \printchaptername을 사용하지 않고 \prechapternum과 \postchapternum을 정의해주어야 한다. 여기서는 모두 비웠다. ■

Lower level divisions

절 타이틀은 소문자의 small cap이고 본문과 같은 크기의 폰트이다. 타이틀에 번호가 붙는데, 장과 절 번호를 모두 포함한다.

다음과 같이 정의하였다.

```
\setsecheadstyle{\raggedright\scshape\MakeLowercase}
  \setbeforesecskip{-\onelineskip}
  \setaftersecskip{\onelineskip}
```

\setsecheadstyle 매크로는 타이틀을 소문자로 바꾼 다음 smallcap으로 식자한다.

타이틀 전후의 간격은 가변길이가 기본값이다. 그러나 이것은 텍스트의 특정 행에 라인을 맞출려할 때는 잘 작동하지 않는다. 섹션 타이틀이 있으면 수직 간격이 조금씩 조절되어 라인업이 어긋나게 된다. 그래서 가변길이 대신 고정길이를 쓰기로 하였다. 타이틀 앞(\setbeforesecskip)과 뒤(\setaftersecskip)에 빈 줄 하나씩을 넣었다.

subsection 타이틀은 이 책에서 가장 하위 장절명령이고 본문 글꼴의 이탤릭체를 사용한다. 번호붙은 들여쓰기 없는 문단으로 식자된다. 여기에 해당하는 코드는 다음과 같다.

```
\setsubsecheadstyle{\sethangfrom{\noindent ##1}\raggedright\itshape}
  \setbeforesubsecskip{-\onelineskip}
  \setaftersubsecskip{\onelineskip}
```

\section 스타일의 경우와 마찬가지로 타이틀 텍스트의 전후에 고정 길이를 넣었다. 타이틀은 normal 사이즈의 italic 폰트로 raggedright 식자된다(setsubsecheadstyle). \sethangfrom 매크로는 내부 매크로 \@hangfrom을 재정의하여 타이틀과 번호가 기본값인 내어쓰기 문단이 아니라 블럭문단으로 식자되도록 한다. \@hangfrom을 재정의할 때 인자를 지정하는 위치에 ##와 같이 두 개의 표시를 쓰고 있다는 점에 주의하라.

18.5 SPECIFYING THE PAGESTYLE

페이지 스타일은 이 작업에서 가장 재미있는 부분일 것이다. 장절 타이틀이 페이지의 상단에 놓이지 않고 편집영역의 위에서 일곱번째 줄 부분 마진에 놓인다. 페이지 번호는

페이지 바닥에 편집영역의 바깥쪽에 정렬되어 놓인다.

페이지 번호를 두는 것은 쉽다. 그것부터 처리하자.

```
%% Bringhurst page style
\makepagestyle{bringhurst}
\makeevenfoot{bringhurst}{\thepage}{}{}
\makeoddfoot{bringhurst}{}{}{\thepage}
```

Specifying the pagestyle

페이지의 특정 위치에 텍스트를 두는 것은 먼저 그 텍스트를 0-width 그림(적어도 LaTeX이 인식하기로는 0의 폭을 갖는 그림) 안에 집어넣는 것이다. 그런 다음 그 그림을 페이지의 특정 위치에 가져다 둔다. 헤더에다가 매다는 방법으로 할 수 있다.

우리는 이 타이틀이 마진 노트에 대해서도 정렬되게 하려 한다. 즉, 마진에서 `\marginparsep` (17pt) 만큼 떨어지도록 하고 폭을 `\marginparwidth` (51pt)로 할 것이다. 이 사용안내서의 앞 장에서 이미 `\pwlayi`와 `\pwlayii`라는 두 개의 길이를 정의했는데 아직까지 사용되지 않았다. 이것을 임시 길이변수로 해서 필요한 계산을 수행하려 한다.

홀수쪽 페이지 헤더에서 그림은 헤더의 ⟨*right*⟩ 쪽에 오고 짝수쪽 페이지에서는 그림이 헤더의 ⟨*left*⟩ 쪽에 온다. 기타 부분은 모두 비운다.

⟨*right*⟩ 쪽에 오는 그림은 기준선에서 오른쪽으로 17pt 떨어져야 한다. 그리고 기준점에서 아래로도 얼마간 띄워야 한다. 이 띄울 거리는 `\headsep`에 `\topskip`을 더한 다음 다시 7.3행을 더해주면 되는 것임을 시행착오 끝에 알아낼 수 있다. 이것은 다음과 같이 계산한다.

```
\setlength{\pwlayi}{\headsep}
\addtolength{\pwlayi}{\topskip}
\addtolength{\pwlayi}{7.3\onelineskip}
```

`\strip@pt`라는 조그마한 LaTeX 내부 매크로가 있다. 한 번도 들어본 적이 없을 것이다. 나도 아주 최근에야 알게 되었다. 이것이 하는 일은 길이값에서 'pt'를 삭제하고 그 값을 보통의 수로 바꾸어주는 것이다. `\unitlength` 기본값은 1pt라는 점을 명심하고 다음을 보자.

```
\makeatletter
\newcommand{\bringpicr}[1]{%
  \setlength{\unitlength}{1pt}
  \begin{picture}(0,0)
    \put(\strip@pt\marginparsep, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedright\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

`\bringpicr{`⟨*text*⟩`}`라는 새로운 매크로는 ⟨*text*⟩를 `\marginparwidth` 폭의 `minipage`에 넣은 다음 raggedright에 이탤릭 폰트로 식자하고, 이 minipage의 좌상귀 점을 (`\marginparsep`, `-\pwlayi`) 위치에 zero-width 그림으로 놓는다.

⟨*left*⟩ 위치에 올 그림은 편집영역의 왼쪽끝에서 17pt 떨어진 곳에 raggedleft로 식자해야 한다. `\pwlayii` 길이변수를 이용해서 `\marginparsep`과 `\marginparwidth`를 더하는 계산을 수행하겠다.

```
\makeatletter
\setlength{\pwlayii}{\marginparsep}
\addtolength{\pwlayii}{\marginparwidth}
\newcommand{\bringpicl}[1]{%
  \setlength{\unitlength}{1pt}
  \begin{picture}(0,0)
    \put(-\strip@pt\pwlayii, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedleft\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

An example design

새로운 매크로 `\bringpicl{`⟨*text*⟩`}`는 ⟨*text*⟩를 `\marginparwidth` 폭의 `minipage`에 넣고 이탤릭체로 raggedleft 식자한 다음, 이 minipage의 좌상귀를 (`-(\marginparsep+\marginparwidth)`, `-\pwlayi`) 위치에 가져다둔다.

이제 페이지 스타일 설정의 남은 부분을 진행하자. 다음 부분은 chapter와 section 타이들을 `\...mark` 매크로에 넣는 것이다.

```
\makeatletter
\makepsmarks{bringhurst}{%
  \let\@mkboth\markboth
  \def\chaptermark##1{\markboth{##1}{##1}}
  \def\sectionmark##1{\markright{##1}}
}
\makeatother
```

마지막으로, 짝수쪽 헤더에 `\bringpicl`을 이용하여 section 타이틀을 표시하고, 홀수쪽 헤더에 `\bringpicr`의 인자로 chapter title을 주도록 한다.

```
\makeevenhead{bringhurst}{\bringpicl{\rightmark}}{}{}
\makeoddhead{bringhurst}{}{}{\bringpicr{\leftmark}}
```

18.6 CAPTIONS AND THE TOC

그림과 표의 캡션은 small sans 폰트로 식자되고 캡션 이름과 번호는 붙지 않는다. LoF나 LoT도 없다. 캡션 타이틀을 원하는 폰트로 설정하는 것은 간단하다.

```
\captiontitlefont{\small\sffamily}
```

표와 그림에 캡션을 붙이는 데는 두 가지 옵션이 있다. 하나는 \legend 명령(무기명의 번호없는 타이틀)을 쓰는 방법이고 다른 하나는 \caption 명령을 수정하여 사용하는 것이다. 디자인이 나중에 변경되어서 캡션 번호를 붙여야 할 경우가 있을지 모르므로 \caption을 수정해서 사용하는 것이 좋은 방법이다. 이 경우에는 매우 간단한데, 그저 \caption 명령에 \legend 명령의 정의를 동일하게 지정해주면 되는 것이다.

```
\let\caption\legend
```

Preamble or package?

잠깐 다른 얘기를 하자면, 처음에 우리는 292 쪽에 있는 그림에서 기본 캡션 스타일(블럭 문단)을 사용하였다. 그러나 이 경우에는 마지막 행이 중앙정렬되어야 하므로 부적절해보인다. 다른 환경과 같이 플로트 환경도 그룹을 이루기 때문에 환경 범위 안에서 변경을 가하는 것이 가능하다. 여기서는 이렇게 했다.

```
\begin{figure}
\captiontitlefont{\small\sffamily}
\captionstyle{\centerlastline}
...
\legend{...} \label{...}
\end{figure}
```

제대로 하려면 캡션 스타일을 바꾸어야 할 것이다. 기본 스타일은 한 줄짜리 캡션에서는 잘 작동하지만, 두세 줄을 가진 경우에는 centering이나 centerlastline 스타일이 더 낫다. 아주 긴 캡션의 경우라면 오히려 블럭 캡션 스타일이 가장 좋다.

chapter 타이틀만이 ToC에 포함된다. 이것은 \settodepth 명령으로 지시한다.

```
\settocdepth{chapter}
```

ToC는 안내선 없이 페이지 숫자가 장 타이틀 바로 뒤에 이어서 나오도록 하고 raggedright로 식자된다. 이것은 다음과 같이 지정한다.

```
\renewcommand{\cftchapterfont}{\normalfont}
\renewcommand{\cftchapterpagefont}{\normalfont}
\renewcommand{\cftchapterpresnum}{\bfseries}
\renewcommand{\cftchapterleader}{}
\renewcommand{\cftchapterafterpnum}{\cftparfillskip}
```

18.7 PREAMBLE OR PACKAGE?

문서 스타일을 변경하거나 새로운 매크로를 몇 개 정의하게 되면, 이 변경을 어디에 두어야 하는가, 즉 문서의 preamble에 둘 것인지 별도의 패키지로 할 것인지에 대해 의문이 생긴다.

만약 이 변경이나 매크로를 하나 이상의 문서에서 사용할 생각이라면 패키지로 만드는 것을 추천한다. 특정 단일 문서에서만 사용할 것이라면 아무래도 상관없다.

이 장에서 보여준 코드는 preamble에 두는 경우를 상정해서 \makeatletter, \makeatother로 @ 문자 있는 매크로의 주변을 둘러쌌다. 이 코드는 예컨대 bringhurst라는 이름의 패키지로 쉽게 만들 수 있다. 즉, 모든 코드를 집어넣고 \makeatletter와 \makeatother

명령만 제외하여 `bringhurst.sty`라는 이름의 파일로 저장하면 되는 것이다. 이 파일의 마지막에 `\endinput`을 추가하는 것도 좋다. LaTeX은 파일을 읽어들이다가 이 명령을 만나면 그 이후는 모두 무시한다.

이제 bringhurst 패키지를 다른 패키지와 마찬가지로,

```
\usepackage{bringhurst}
```

문서의 preamble에 위와 같이 써주면 된다.

An example design

부 록

부록 A 패키지와 매크로

이 장은 *section* 장 스타일로 식자한다. 그 외의 부분은 제 18 장에 정의한 레이아웃을 그대로 이용한다.

> 【memhangul】 memhangul-ucs 패키지는 한글 문서의 특성상 부록장에 대해서 본문장의 스타일을 그대로 적용하기 어려운 경우 `appendixsection`과 같은 페이지 스타일을 별도로 제공한다. 번역본은 이 장 스타일을 사용하였다. ■

1 서론

memoir 클래스가 이 매뉴얼에서 본 모든 것을 제공해주지는 않는다. 저자는 일반적인 LaTeX 배포판에서 찾을 수 있는 패키지 몇 가지를 사용하였다. 그리고 새로운 매크로 몇 가지를 추가로 정의하였다.

2 패키지(PACKAGES)

일반적으로 사용자 시스템에 설치되어 있을 것으로 생각되는, 이 글에서 사용한 패키지는 다음과 같다. 만약 이 패키지들이 없다면 설치하는 것이 좋을 것이다.

- url [Ars99] 패키지는 URL을 특수문자나 행나눔에 신경쓰지 않고 식자할 수 있게 해준다.
- fixltx2e [MC00] 패키지는 오리지널 LaTeX 커널의 몇 가지 문제점을 제거해준다. 특히 2단 문서에서 플로트의 순서를 유지해주는 것과 역시 2단 문서에서 `\leftmark`와 `\rightmark`가 풀리는 것을 방지해준다.
- alltt [Bra97] 패키지는 verbatim-류 환경과 비슷하지만 `\`, `{`, `}` 문자들이 원래의 의미대로 사용되게 함으로써 LaTeX 명령을 쓸 수 있게 하는 기본 패키지이다.
- graphicx [CR99] 패키지는 여러 가지 그림 관련 기능을 제공하는 필수 패키지이다.

여기에서 사용되었지만 보통 가지고 있지 않은 경우가 많은 것은 layouts [Wil99a] 패키지이다. 레이아웃 다이어그램을 그리는 데 사용하였고, 그림 (8.1)와 그림 (8.2) 등에서 그 결과를 볼 수 있다. 이런 그림을 다음과 같은 간단한 코드로 그리게 한다.

```
\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\drawheading{}
\caption{Displayed sectional headings} \label{fig:displaysechead}
\end{figure}

\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\runinheadtrue
\drawheading{}
\caption{Runin sectional headings} \label{fig:runsechead}
\end{figure}
```

또 이 패키지는 여러 레이아웃 파리미터 값을 주어서 결과가 어떻게 변하는지를 그림으로 그려보일 수 있다.

이 사용자 안내서에서 사용된 layouts의 버전은 2001/04/30 날짜의 v2.4이다. 이 이전 버전은 몇 가지 그림, 예컨대 그림 (6.2)와 같은 그림을 그리지 못한다.

3 매크로(MACROS)

이 매뉴얼의 preamble은 많은 매크로 정의를 포함하고 있다. 일반적인 문서에서 이렇게 까지 많은 정의가 필요하지는 않을 것인데, 그 까닭은,

- 여러 LaTeX 명령을 직접 식자해야 했기 때문이다. 이를 위해서는 약간의 특별한 처리가 필요했다.
- 이 매뉴얼이 성공적으로 LaTeX 처리되게 하는 데 최소한의 외부 패키지를 필요로 하도록 만들려 했기 때문이다. 그래서 여러 매크로들을 다른 곳에서 베껴왔다.
- 자동 색인을 구현하고 싶었기 때문이다.
- 본문에서 구문 설정과 예제 코드를 보이고 싶었기 때문이다.

매크로 정의를 확인하고 싶으면 제일 좋은 방법은 preamble을 직접 읽는 것이다. 그렇지만 일반적으로 관심이 갈 것으로 보이는 몇 가지를 아래에서 보이겠다.

`\pstyle{`⟨*style*⟩`}`

`\pstyle` 명령은 사용된 페이지 스타일을 나타내는 인자를 기울인 글꼴(slanted font)로 식자하고 페이지 스타일 엔트리를 색인에 넣는다. 정의는 다음과 같이 되어 있다.

```
\newcommand{\pstyle}[1]{\textsl{#1}\index{#1pages@\textsl{#1} (pagestyle)}}
```

첫번째 부분은 인자를 텍스트로 프린트하는 것이고 두번째 부분은 엔트리를 `.idx`에 쓰는 것이다. `#1pages` 부분은 MAKEINDEX 프로그램이 나중에 엔트리를 정렬할 때 사용한다. `@` 문자 이후 부분이 색인에 들어간다.

패키지와 매크로

`\cstyle{`⟨*style*⟩`}`

`\cstyle` 명령은 chapterstyle에 해당하는 그 인자를 기울인 글꼴로 식자하고 역시 색인에 chapterstyle 엔트리로 넣는다. 정의는 다음과 같다.

```
\newcommand{\cstyle}[1]{\textsl{#1}\index{#1chaps@\textsl{#1} (chapterstyle)}}
```

이것은 `\pstyle`의 경우와 거의 같다.

*companion*이라는 chapterstyle도 있고 *companion*이라는 pagestyle도 있다. 색인을 정렬할 때 쓰는 문자열은 각각 `companionchaps`와 `companionpages`이기 때문에, 장 스타일이 페이지 스타일보다 색인에서 먼저 나온다. 정렬에 쓰이는 문자열과 실제 엔트리 문자열을 다르게 하는 주된 이유는 다른 종류의 엔트리지만 같은 이름을 가진 경우를 구별하기 위한 이유도 있고, 정렬 순서가 형식 명령에 의해 혼선을 빚을 가능성을 피하기 위해서이기도 하다.

`\begin{syntax}` syntax `\end{syntax}`

`syntax` 환경은 명령과 환경의 구문법을 보여주기 위한 것이다. 정의는 다음과 같다.

```
\newenvironment{syntax}{\begin{center}
                        \begin{tabular}{|p{0.9\linewidth}|} \hline}%
                       {\hline
                        \end{tabular}
                        \end{center}}
```

이 명령은 `tabular` 환경을 이용하여 구현되었다. 이 환경은 페이지 나눔이 일어나지 않는 박스로 이루어져 있다. 박스의 프레임은 평범한 가로선과 세로선으로 `tabular` 환경에서 얻을 수 있는 것들이다. 폭은 `\textwidth`의 90%로 고정되어 있다. 이것이 `tabular` 환경으로 이루어졌기 때문에, 각 행은 `\\`로 끝나야 한다. `syntax` 환경 안에서는 일반적인 LaTeX 명령이 작동하므로, 그 안에 원하는 것을 놓는 것이 쉽게 이루어진다.

`\begin{lcode}` LaTeX code `\end{lcode}`

lcode 환경은 LaTeX 코드 예제를 보여주기 위해 사용된다. 이것은 verbatim 환경의 특별한 경우로서 폰트 사이즈가 \small이고 행간이 \baselineskip으로 되어 있다. 각 행은 들여쓰기가 된다. 이 환경의 정의에는 verbatim 패키지가 이용되었다.

이 환경의 마지막 부분에는 코드에서 명시적으로 드러나지는 않지만 list가 사용되고 있다. 자세한 것은 verbatim 패키지 문서 [SRR99]를 참고하라. 이 환경을 tight list로 만들기 위해서 두 개의 보조적 item을 정의하였다.

매크로
(Macros)

```
% \@zeroseps sets list before/after skips to minimum values
\newcommand{\@zeroseps}{\setlength{\topsep}{\z@}
                        \setlength{\partopsep}{\z@}
                        \setlength{\parskip}{\z@}}
% \gparindent is the \parindent for the body text
\newlength{\gparindent} \setlength{\gparindent}{\parindent}
```

\@zeroseps 매크로는 리스트의 전, 후, 중간 skip 값을 0pt (\z@는 0pt를 줄여쓴 것이다)로 설정한다. \parindent 값을 \gparindent에 저장한다. 이 값은 환경의 행 들여쓰기에 적용된다.

```
% Now we can do the new lcode verbatim environment.
% This has no extra before/after spacing.
\newenvironment{lcode}{\@zeroseps
  \renewcommand{\verbatim@startline}%
    {\verbatim@line{\hskip\gparindent}}
  \small\setlength{\baselineskip}{\onelineskip}\verbatim}%
  {\endverbatim
   \vspace{-\baselineskip}\noindent
}
```

{\hskip\gparindent} 부분은 \gparindent 간격만큼을 각 행 앞에 붙인다.

\small\setlength{\baselineskip}{\onelineskip} 부분은 폰트 사이즈를 \small로 한다. 이렇게 하면 \baselineskip 값이 normal font에 비하여 줄어들게 되는데, 이것은 \baselineskip을 임시로 normal skip인 \onelineskip으로 바꿈으로써 수정된다. 이 환경의 끝에는 LaTeX이 집어넣는 한 줄을 삭제하기 위한 한 줄 만큼의 음수값이 붙어 있다.

§2.4에 있는 두 개의 문단머릿글자는 preamble에 정의된 매크로에 의하여 식자되었다. 두 개 중 첫번째의 좀 빈약한 것은 \versal 매크로를 이용한 것이다. 두번째 것은 \drop 매크로를 이용하였는데, 이 매크로는 1998년에 David Cantor와 Dominik Wujastyk이 LaTeX 2.09를 위하여 작성한 것이었다. 현재는 CTAN에 좀더 많은 패키지들이 있다. lettrine 패키지 [Fli98]가 쓸 만했다.

참고 문헌

CTAN은 Comrehensive TeX Archive Network이다. CTAN에 접속하고 이용하는 방법에 대해서는 http://www.tug.org를 참고하라.

[Ado01] *How to Create Adobe PDF eBooks*. Adobe Systems Inc., 2001. (Available from http://www.adobe.com/epaper/tips/acr5ebook/pdfs/eBook.pdf)

[Ars99] Donald Arseneau. *The url package*. February, 1999. (Available from CTAN as /macros/latex/contrib/misc/url.sty)

[Ars01a] Donald Arseneau. *The titleref package*. April, 2001. (Available from CTAN as /macros/latex/contrib/misc/titleref.sty)

[Ars01b] Donald Arseneau. *The framed package*. July, 2001. (Available from CTAN as /macros/latex/contrib/misc/framed.sty)

[Ars01b] Donald Arseneau. *The chapterbib package*. September, 2001. (Available from CTAN as /macros/latex/contrib/cite/chapterbib.sty)

[Bar92] Helen Barolini. *Aldus and his Dream Book*. Italica Press (ISBN 0–934977–22–4), 1992.

[BDG89] Charles Bigelow, Paul Hayden Duensing and Linnea Gentry (Eds). *Fine Print on Type*. 1989. Fine Print, CA (ISBN 0–9607290-X) or Bedford Arts, CA (ISBN 0–938491–17–2).

[Boh90] Robert Bohle. *Publication Design for Editors*. Prentice-Hall, 1990.

[Ber02] Jens Berger. *The titlesec and titletoc packages*. September, 2002. (Available from CTAN in /macros/latex/contrib/titlesec)

[Bez99] Javier Bezos. *The titlesec and titletoc packages*. February, 1999. (Available from CTAN in /macros/latex/contrib/titlesec)

[Bra94] Johannes Braams *et al.* *Standard LaTeX2e packages makeidx and showidx.* November, 1994. (Available from CTAN as /macros/latex/base/makeindx.dtx(ins))

[Bra97] Johannes Braams. *The alltt environment.* June, 1997. (Available from CTAN as /macros/latex/base/alltt.dtx(ins))

[Bri92] Robert Bringhurst. *The Elements of Typographic Style.* Hartley & Marks (ISBN 0–88179–033–8), 1992.

[Bur59] C. L. Burt. *A Psychological Study of Typography.* Cambridge University Press, 1959.

[Car94] David Carlisle. *The delarray package.* March, 1994. (Available from CTAN in /macros/latex/required/tools)

[Car95] David Carlisle. *The afterpage package.* October, 1995. (Available from CTAN in /macros/latex/required/tools)

[Car98a] David Carlisle. *The color package.* May, 1998. (Available from CTAN in /macros/latex/required/tools)

[Car98b] David Carlisle. *The longtable package.* May, 1998. (Available from CTAN in /macros/latex/required/tools)

[Car98c] David Carlisle. *The enumerate package.* August, 1998. (Available from CTAN in /macros/latex/required/tools)

[Car98d] David Carlisle. *The remreset package.* August, 1998. (Available from CTAN in /macros/latex/contrib/carlisle)

[Car99a] David Carlisle. *The tabularx package.* January, 1999. (Available from CTAN in /macros/latex/required/tools)

[Car01] David Carlisle. *The dcolumn package.* May, 2001. (Available from CTAN in /macros/latex/required/tools)

[CR99] David Carlisle and Sebastian Rahtz. *The graphicx package.* February, 1999. (Available from CTAN in /macros/latex/required/graphics)

[CB99] Warren Chappell and Robert Bringhurst. *A Short History of the Printed Word.* Hartley & Marks, 1999. ISBN 0–88179–154–7.

[CH88] Pehong Chen and Michael A. Harrison. 'Index Preparation and Processing'. *Software: Practice and Experience*, 19:8, pp. 897–915, September, 1988. (Available from CTAN in /indexing/makeindex/paper)

[Chi93] *The Chicago Manual of Style*, Fourteenth Edition. The University of Chicago (ISBN 0–226–10389–7) 1993.

[Coc02] Steven Douglas Cochran. *The subfigure package*. March, 2002. (Available from CTAN in /macros/latex/contrib/subfigure)

[CG96] John H. Conway and Richard K. Guy. *The Book of Numbers*. Copernicus, Springer-Verlag (ISBN 0–387–97993–X), 1996.

[Cra92] James Craig. *Designing with Type: A Basic Course in Typography*. Watson-Guptill, NY, 1992.

[Dal99] Patrick W. Daly. *Natural Sciences Citations and References*. May, 1999. (Available from CTAN in /macros/latex/contrib/natbib)

[Deg92] Asaf Degani. *On the Typography of Flight-Deck Documentation*. NASA Contractor Report # 177605. December, 1992. (Available from http://members.aol.com/willadams/typgrphy.htm#NASA)

[Dow96] Geoffrey Dowding. *Finer Points in the Spacing & Arrangement of Type*. Hartley & Marks (ISBN 0–88179–119–9), 1996.

[Dow98] Geoffrey Dowding. *An Introduction to the History of Printing Types*. The British Library and Oak Knoll Press (ISBN 0–7123–4563–9 UK, 1–884718–44–2 USA), 1998.

[Dow00] Michael J. Downes. *The patchcmd package*. July, 2000. (Available from CTAN in /macros/latex/contrib/patchcmd)

[Eij92] Victor Eijkhout. *TeX by Topic*. Addison-Wesley, 1992. ISBN 0–201–56882–9. (Available from http://www.eijkhout.net/tbt/).

[Fai00] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX2e*. March, 2000. (Available from CTAN in /macros/latex/contrib/footmisc)

[Fai98] Robin Fairbairns. *The moreverb package*. December, 1998. (Available from CTAN in /macros/latex/contrib/moreverb)

[FAQ] Robin Fairbairns. *The UK TeX FAQ*. (Available from CTAN in /help/uk-tex-faq)

[Fea03] Simon Fear. *Publication quality tables in LaTeX*. March, 2003. (Available from CTAN in /macros/latex/contrib/booktabs)

[Fli98] Daniel Flipo. *Typesetting 'lettrines' in LaTeX2e documents*. March, 1998. (Available from CTAN in /macros/latex/contrib/lettrine)

[Fra00] Melchior Franz. *The crop package*. February, 2000. (Available from CTAN in /macros/latex/contrib/crop)

[FOS98] Friedrich Friedl, Nicolaus Ott and Bernard Stein. *Typography: An Encyclopedic Survey of Type Designs and Techniques throughout History*. Black Dog & Leventhal Publishers Inc. (ISBN 1–57912–023–7), 1998.

[Gar66] Martin Gardner. *More Mathematical Puzzles and Diversions*. Penguin Books (ISBN 0–14–020748–1), 1966.

[GMS94] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley Publishing Company (ISBN 0–201–54199–8), 1994.

[GRM97] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript*. Addison-Wesley Publishing Company (ISBN 0–201–85469–4), 1997.

[GR99] Michel Goossens and Sebastian Rahtz (with Eitan Gurari, Ross Moore and Robert Sutor). *The LaTeX Web Companion: Integrating TeX, HTML and XML*. Addison-Wesley Publishing Company (ISBN 0–201–43311–7), 1999.

[Gou87] J. D. Gould *et al*. 'Reading from CRT displays can be as fast as reading from paper'. *Human Factors*, pp 497–517, 29:5, 1987.

[HR83] J. Hartley and D. Rooum. 'Sir Cyril Burt and typography'. *British Journal of Psychology*, pp 203–212, 74:2, 1983.

[HM01] Steven Heller and Philip B. Maggs (Eds). *Texts on Type: Critical Writings on Typography*. Allworth Press (ISBN 1–58115–082–2), 2001.

[Hoe98] Alan Hoenig. *TeX Unbound: LaTeX and TeX strategies for fonts, graphics, and more*. Oxford University Press (ISBN 0–19–509686–X), 1998.

[HK75] J. K. Hvistendahl and M. R. Kahl. 'Roman vs. sans serif body type: Readability and reader prference'. *AANPA News Research Bulletin*, pp 3–11, 17 Jan., 1975.

[Jon95] David M. Jones. *A new implementation of LaTeX's indexing commands*. September, 1995. (Available from CTAN in /macros/latex/contrib/camel)

[Knu84] Donald E. Knuth. *The TeXbook*. Addison-Wesley Publishing Company (ISBN 0–201–13448–9), 1984.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Publishing Company (ISBN 0–201–52983–1), 1994.

[LMB99] Leslie Lamport, Frank Mittelbach and Johannes Braams. *Standard document classes for LaTeX version 2e*. September, 1999. (Available from CTAN as /macros/latex/base/classes.dtx)

[Law90] Alexander Lawson. *Anatomy of a Typeface*. David R. Godine (ISBN 0–87923–333–8), 1990.

[Leu92] Mary-Claire van Leunen. *A Handbook for Scholars*. Oxford University Press (ISBN 0–19–506954–4), 1992.

[Lon91] F. W. Long. *multind*. August, 1991. (Available from CTAN as /macros/latex209/contrib/misc/multind.sty)

[McD98] Rowland McDonnell. *The sectsty package*. November, 1998. (Available from CTAN in /macros/latex/contrib/secsty)

[McL75] Ruari McLean. *Jan Tschichold: Typographer*. David R. Godine (ISBN 0–87923–841–0), 1975.

[McL80] Ruari McLean. *The Thames & Hudson Manual of Typography*. Thames & Hudson (ISBN 0–500–68022–1), 1980.

[McL95] Ruari McLean (Ed). *Typographers on Type*. W. W. Norton & Co. (ISBN 0–393–70201–4), 1995.

[Mit95] Frank Mittelbach. *The doc and shortvrb packages*. May, 1995. (Available from CTAN in /macros/latex/base)

[Mit95] Frank Mittelbach. *The doc and shortvrb packages*. May, 1995. (Available from CTAN in /macros/latex/base)

[Mit98] Frank Mittelbach. *An environment for multicolumn output*. January, 1998. (Available from CTAN in /macros/latex/required/tools)

[MC98] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment*. May, 1998. (Available from CTAN in /macros/latex/required/tools)

[MC00] Frank Mittelbach and David Carlisle. *The fixltx2e package*. September, 2000. (Available from CTAN in /macros/latex/base)

[Mor99] Stanley Morison. *A Tally of Types*. David R. Godine (ISBN 1–56792–004–7), 1999.

[NG98] Rolf Niespraschk and Hubert Gäßlein. *The sidecap package*. June, 1998. (Available from CTAN in /macros/latex/contrib/sidecap)

[Oet] Tobias Oetiker. *The Not So Short Introduction to LaTeX2e*. (Available from CTAN in /info/lshort/english)

[Oos96] Piet van Oostrum. *Page Layout in LaTeX*. June, 1996. (Available from CTAN in /macros/latex/contrib/fancyhdr)

[Pak01] Scott Pakin. *The Comprehensive LaTeX Symbol List*. July, 2001. (Available from CTAN in /info/symbols/comprehensive)

[Pug02] Diego Puga. *The Pazo Math fonts for mathematical typesetting with the Palatino fonts*. May, 2002. (Available from CTAN in /fonts/mathpazo)

[Rahtz01] Sebastian Rahtz. *Section name references in LaTeX*. January, 2001. (Available from CTAN in /macros/latex/contrib/hyperref)

[Rahtz02] Sebastian Rahtz. *Hypertext marks in LaTeX*. May, 2002. (Available from CTAN in /macros/latex/contrib/hyperref)

[Rec97] Keith Reckdahl. *Using Imported Graphics in LaTeX2e*. December, 1997. (Available from CTAN as /info/epspatex.ps or /info/epslatex.pdf)

[Reh72] Rolf Rehe. 'Type and how to make it most legible'. *Design Research International*, 1972.

[RAE71] D. O. Robinson, M. Abbamonte and S. H. Evans. 'Why serifs are important: The perception of small print'. *Visible Language*, pp 353–359, 4, 1971.

[Rog43] Bruce Rogers. *Paragraphs on Printing*. William E. Rudge's Sons, Inc. (no ISBN), 1943. (Reissued by Dover, 1979, ISBN 0–486–23817–2)

[Rog49] Bruce Rogers. *Centaur Types*. October House (no ISBN), 1949.

[SW94] Douglas Schenck and Peter Wilson. *Information Modeling the EXPRESS Way*. Oxford University Press (ISBN 0–19–508714–3), 1994.

[SRR99] Rainer Schöpf, Bernd Raichle and Chris Rowley. *A New Implementation of LaTeX's verbatim and verbatim* Environments*. December, 1999. (Available from CTAN in /macros/latex/required/tools)

[Sch97] Karen A. Schriver. *Dynamics in Document Design*. Wiley & Sons, 1997.

[Thi99] Christina Thiele. 'The Treasure Chest: Package tours from CTAN', *TUGboat*, vol. 20, no. 1, pp 53–58, March 1999.

[Tin63] Miles A. Tinker. *Legibility of Print*. Books on Demand (University Microfilms International), 1963.

[Tsc91] Jan Tschichold. *The Form of the Book*. Lund Humphries (ISBN 0–85331–623–6), 1991.

[Ume99] Hideo Umeki. *The geometry package*. November, 1999. (Available from CTAN in /macros/latex/contrib/geometry)

[Whe95] Colin Wheildon. *Type & Layout*. Strathmore Press (ISBN 0–9624891–5–8), 1995.

[Wil93] Adrian Wilson. *The Design of Books*. Chronicle Books (ISBN 0–8118–0304–X), 1993.

[Wil99a] Peter Wilson. *The layouts package*. January, 1999. (Available from CTAN in /macros/latex/contrib/layouts)

[Wil99b] Peter Wilson. *The tocvsec2 package*. January, 1999. (Available from CTAN in /macros/latex/contrib/tocvsec2)

[Wil00a] Peter Wilson. *The epigraph package*. February, 2000. (Available from CTAN in /macros/latex/contrib/epigraph)

[Wil00b] Peter Wilson. *LaTeX files for typesetting ISO standards*. February, 2000. (Available from CTAN in /macros/latex/contrib/isostds/iso)

[Wil00c] Peter Wilson. *The nextpage package*. February, 2000. (Available from CTAN as /macros/latex/contrib/misc/nextpage.sty)

[Wil00d] Peter Wilson. *The needspace package*. March, 2000. (Available from CTAN as /macros/latex/contrib/misc/needspace.sty)

[Wil00e] Peter Wilson. *The xtab package*. April 2000. (Available from CTAN in macros/latex/contrib/xtab)

[Wil01a] Peter Wilson. *The abstract package*. February, 2001. (Available from CTAN in /macros/latex/contrib/abstract)

[Wil01b] Peter Wilson. *The chngpage package*. February, 2001. (Available from CTAN as /macros/latex/contrib/misc/chngpage.sty)

[Wil01c] Peter Wilson. *The appendix package*. March, 2001. (Available from CTAN in /macros/latex/contrib/appendix)

[Wil01d] Peter Wilson. *The ccaption package*. March, 2001. (Available from CTAN in /macros/latex/contrib/ccaption)

[Wil01e] Peter Wilson. *The chngcntr package*. April, 2001. (Available from CTAN as /macros/latex/contrib/misc/chngcntr.sty)

[Wil01f] Peter Wilson. *The hanging package*. March, 2001. (Available from CTAN in /macros/latex/contrib/hanging)

[Wil01g] Peter Wilson. *The titling package*. March, 2001. (Available from CTAN in /macros/latex/contrib/titling)

[Wil01h] Peter Wilson. *The tocbibind package*. April, 2001. (Available from CTAN in /macros/latex/contrib/tocbibind)

[Wil01i] Peter Wilson. *The tocloft package*. April, 2001. (Available from CTAN in /macros/latex/contrib/tocloft)

[Wil01j] Peter Wilson. *The LaTeX memoir class for configurable book typesetting: Source code*. July, 2001. (Available from CTAN in /macros/latex/contrib/memoir)

[Wil01k] Peter Wilson. *Typesetting simple verse with LaTeX* July, 2001. (Available from CTAN in /macros/latex/contrib/verse)

[Wil01l] Peter Wilson. *Printing booklets with LaTeX* August, 2001. (Available from CTAN in /macros/latex/contrib/booklet)

[Wil03] Peter Wilson. *ledmac: A presumptuous attaempt to port EDMAC and TABMAC to LaTeX* November, 2003. (Available from CTAN in /macros/latex/contrib/ledmac)

[Wil??] Peter Wilson. *A Rumour of Humour: A scientist's commonplace book*. To be published?

[Zac69] B. Zachrissom. *Studies in the Legibility of Printed Text*. Almqvist & Wiksell, Stockholm, 1969.

[Zap00] Hermann Zapf. *The Fine Art of Letters*. The Grolier Club (ISBN 0–910672–35–0), 2000.

찾아보기

색인 항목의 첫번째로 언급된 페이지가 대체로 그 주제에 대한 주요 참조 위치이지만 항상 그런 것은 아니다.

Colophon

This manual was typeset using the LaTeX typesetting system created by Leslie Lamport and the memoir class. The body text is set 10/12pt on a 33pc measure with Computer Modern Roman designed by Donald Knuth. Other fonts include Sans, Smallcaps, Italic, Slanted and Typewriter, all from Knuth's Computer Modern family.

이 매뉴얼은 Leslie Lamport의 LaTeX 문서준비시스템과 Peter Wilson의 memoir 클래스, 그리고 memhangul-ucs 패키지를 이용하여 조판하였다. 본문의 라틴 문자는 Donald Knuth가 디자인한 Computer Modern Roman 및 Sans, Smallcaps, Italic, Slanted, Typewriter 글꼴을 이용하였고, 한글 문자 식자에는 ko.TeX 기본 글꼴(은글꼴 type 1)을 사용하였다.

이 매뉴얼의 번역은 2005년에 이루어졌던 것인데, 현재 영문판은 새 버전이 나와 있다. 새 버전의 번역이 나올 때까지 참고하도록 이전 문서를 새로 컴파일하여 공개해둔다. (2008/11/05)