The Asian Journal of TEX, Volume 5, No. 1, June 2011
Article revision 2011/6/1

클래스 oblivoir와 책 만들기

Typesetting a book with the oblivoir class

김강수 Kangsoo Kim
한글텍사용자그룹 karnes@ktug.or.kr

Keywords oblivoir, XeTeX, XeTeX-k*o*, typesetting, book

Abstract oblivoir는 memoir 클래스를 이용하여 한글 문서를 작성할 수 있게 한 클래스이다. XeTeX과 ko.TeX을 이용할 수 있게 된 이래 텍을 이용하는 한글 도서의 고품위 조판이 가능하게 되었으나 적절한 안내서가 아직 충분하지 않다. 이 글은 oblivoir를 실제 출판에 이용하기 위해서 어떤 요소가 필요한지를 간단한 한 권의 책을 디자인해봄으로써 개관하고자 한다. 이를 통하여 oblivoir 사용안내서를 작성할 경우 포함되어야 할 내용의 범위를 미리 짐작해보려 한다.

1 들어가는 말

라텍 클래스 oblivoir[1]가 발표된 것은 2006년의 일이다. 이 클래스는 한국어–한글 문헌을 식자 조판하기 위한 범용 클래스로서, memoir 클래스에서 한글을 사용하기 위해 제작된 memhangul-ucs라는 패키지를 바탕으로 필자에 의해 제작되었다 [9]. Peter Wilson이 작성한 memoir는 큰 규모의 도서를 조판하기에 적합하도록 만들어진 것으로 종래부터 이 목적으로 사용되어 오던 book 클래스가 가진 갖가지 문제들을 대부분 해결한 대안 클래스이다 [14].

oblivoir 클래스의 핵심인 memhangul 패키지는 다음과 같은 발전을 거쳤다.[2]

memhangul (2006) 은광희의 한글라텍 HLaTeX으로 한글을 구현하였다. HLaTeX이 가진 문제점 및 유니코드를 지원하는 텍 엔진 또는 매크로 등장 이전의 여러 한계들로 인하여 memoir 위에서 한글 사용이 가능하다는 것을 보여주었다는 정도의 의의를 갖는다.

memhangul-ucs (2006) memhangul의 발전판으로 김도현의 hangul-ucs 패키지를 한글 식자 엔진으로 채택하였다. 유니코드 한글 체계를 채택함으로써 한글 구현상의 많은 애로들을 해결하였다.

memhangul-x (2008) XeTeX과 LuaTeX을 위한 ko.TeX 패키지, XeTeX-k*o*와 LuaTeX-k*o*에 대응하여 새로이 구성된 memhangul로서 현재 권장되고 있는 버전이다.

1. Oblivoir라는 말은 '망각' 이라는 뜻의 oblivion과 memoir를 합친 신조어이다.
2. http://faq.ktug.or.kr/faq/Karnes/Oblivoir

현재 한국어 문헌의 조판을 위해 권장되는 것은 XeTeX 엔진이므로 이 글에서는 XeTeX 및 XeTeX-k*o*, 그리고 memhangul-x를 이용하는 oblivoir만을 다루고자 한다.[3]

필자는 memhangul을 제작하는 과정에서 memoir 매뉴얼 전체를 번역하였고, 이 번역본 [15]는 k*o*.TeX의 일부로 배포되고 있다.[4] memhangul에서 추가하거나 변형한 것, 그리고 한글 문서 작성을 위하여 마련된 것을 첨기하여 memoir 한글 문헌 제작을 위한 매뉴얼로 쓰이기를 기대하였으나 실제 사용에 있어서 이 번역본만으로는 충분하지 않아서 oblivoir 전체에 대한 사용설명서의 필요성이 꾸준히 제기되었다. 그 이유는 다음과 같다.

현재 oblivoir 클래스를 제대로 이용하기 위해서는 적어도 memoir 설명서 [14]와 XeTeX-k*o* 설명서 [5], 그리고 oblivoir 특유의 명령을 해설한 문헌, 특히 [3]을 읽어야 한다. 이 문서들은 일반적인 라텍 사용법을 당연히 전제하고 있기 때문에 초보자가 접근하기에 친절하지 않고 실제로 그 분량이 만만치 않을 뿐 아니라 필요한 기능을 즉시 찾아보기도 어렵다. 게다가 이 문서들에 충분히 설명되지 않거나 문서화되지 않은 기능들이 산재하고 있다. oblivoir 자체가 라텍, memoir, memhangul-x로 이어지는 여러 층위에서 운영되는 클래스이기 때문이기도 하지만 문서를 디자인하거나 작성하려는 사람에게 대단히 불편한 상황이 유지되고 있는 것이다. 이 상황을 타개하기 위해서는 잘 마련된 사용설명서와 참고서가 필요하다.

이 글의 목적은 oblivoir 사용설명서를 작성한다면 어떤 내용이 포함되어야 할는지를 예상해보려는 것이다. 실제로 한 권의 책을 디자인하는 과정을 따라가면서 각 단계에서 어떤 배경이 요구되는가를 기술한다. 모든 주제를 다루지 못하므로 크게 페이지의 구성, 폰트의 활용, 장절표제를 중심으로 이를 살펴보려 한다.

2 문서의 기본 옵션

2.1 책의 크기

책의 크기를 나타내는 데 memoir 클래스는 세 단계의 개념을 구별한다.

용지 (stock) 실제 인쇄가 이루어지는 종이 크기를 말한다. `\stockheight`와 `\stockwidth`로 표시한다.

판형 (paper; page) 재단된 용지를 말하는 것으로 실제 제작이 완료된 책의 크기가 된다. `\paperheight`와 `\paperwidth`로 표시한다.

편집 영역 (typeblock) 상하 여백과 좌우 여백을 제외한 본문이 놓이는 부분을 가리킨다. `\textheight`와 `\textwidth`로 표시한다.

라텍에 익숙한 사용자들도 용지와 판형의 구별에서 당황하는 경우가 있다. 그러나 가만 생각해보면 geometry 패키지에서도 예를 들어 `bindingoffset`과 같은 것을 제공하는데 이 바인딩 오프셋은 판형의 일부일 수 없다. crop 패키지를 쓰면 재단 위치를 표시할 수 있는데

3. memhangul-x의 개발 과정에서 이를 이용하는 oblivoir를 잠정적으로 xoblivoir로 불렀으나 현재는 동작하는 텍 엔진에 따라 oblivoir와 xoblivoir가 자동으로 결정된다.
4. k*o*.TeX이 설치된 컴퓨터의 명령창 또는 터미널에서 '`texdoc memucs`' 명령으로 번역본을 읽을 수 있다.

재단될 자리 바깥쪽도 역시 판형의 일부는 아니다. 그러므로 판형은 용지 위에 놓이고 편집 영역은 판형 위에 놓인다.

일반적으로 논문이나 리포트를 작성할 때 A4 용지를 가장 많이 사용한다. A4 용지는 $\sqrt{2}:1$의 리히텐베르그 비율을 갖는 아름다운 규격이지만 책의 크기로서는 너무 크다는 단점이 있다. 흔히 워드 프로세서들이 그리하듯이 A4 용지에 10mm 내지 20mm 정도의 여백만을 주고 작성한 문서는 글줄의 길이(행장)가 너무 길어서 읽기에 부담스럽고 피곤하다.

라텍 표준 클래스에서 `a4paper` 옵션을 주고 문서를 작성하면 오히려 여백이 상당히 커지는 대신 편집 영역의 폭인 행장은 ──영문 문서의 경우── 적절하게 읽을 만한 길이가 된다. 한글 문서의 경우는 이보다 조금 더 행장이 길어져도 좋을 것 같지만 여전히 '읽기에 좋은' 행장을 선택하면 여백이 너무 커진다고 보아야 할 것이다. 그러므로 A4 용지에서는 면적률이 낮아질 수밖에 없다.

워드 프로세서로 작성한 문서에 익숙해지면 이 '넓은 여백'을 도저히 참지 못하는 경우가 있다. 그리고 전자문서의 경우라면 여백이 넓은 것은 확실히 독서를 방해한다. 논문을 작성해서 제출하는 경우라면 이런 문제에 신경쓸 필요가 없다. 논문을 심사할 기관에서 원하는 크기에 맞추어서 작성하면 될 것이다. 그러나 자신의 책을 디자인하는 경우라면 주의를 기울여야 할 문제다.

판형

책의 판형(크기)을 결정하는 데는 두 가지 요인이 작용한다. 하나는 디자인적인 의도이고 다른 하나는 경제적인 요인이다. 디자인적 의도라 함은 책의 의도를 잘 전달하면서도 아름다움을 잃지 않는 크기를 찾아내는 것이고 경제적 요인이라 함은 규격 용지를 잘라내었을 때 남는 부분의 비율인 손지율[5]을 최소로 하여 제작비를 절감하는 문제와 관련되어 있다.

규격을 고려하여 흔히 쓰이는 책의 판형은 명칭이 같더라도 조금씩 크기가 다른 것이 우리의 출판 현실이다. 판형에 대한 논의는 memoir 클래스의 매뉴얼에서도 충분히 이루어졌고 memhangul-x는 이 문제에 전혀 관계하지 않으므로 우리나라에서 통용되는 일반적인 규격 크기만을 표 1에서 소개하기로 한다.

페이지 디자인

한 페이지는 다음 요소들로 이루어진다.

편집 영역 본문이 놓이는 영역이다. 가장 중요하고 비중이 높을 뿐 아니라 디자인의 핵심적인 대상이 된다. 다단 문서의 경우 편집 영역을 몇 부분으로 분할한다. 그림이나 표도 원칙적으로 편집 영역 안에 와야 한다.

면주 페이지 번호를 표시하는 부분과 장절 표제 등 페이지 전체에 대한 정보를 기록하는 부분으로 상단면주(header)와 하단면주(footer)로 이루어져 있다.

5. 원지를 재단할 때 원지 규격에 맞지 않는 크기를 선택하면 인쇄에 쓰지 못하는 부분이 남을 수밖에 없다. 이 비율을 손지율이라 한다.

표 1. 책의 판형

판형명칭	크기	다른 크기	대응판형	사용종이	용도
국판	148 × 210	152 × 218	A5	국전지	교과서
국배판	210 × 297	218 × 304	A4	국전지	잡지
국반판	105 × 148	109 × 152	A6	국전지	문고
타블로이드	257 × 364	254 × 374	B4	4 × 6전지	정보신문
사륙판	128 × 182	127 × 188	B6	4 × 6전지	문고
사륙배판	182 × 257	188 × 254	B5	4 × 6전지	참고서
신국판	152 × 225	154 × 224	*	국전지	단행본
크라운판	176 × 248	154 × 224	*	4 × 6전지	사진집
30절판	125 × 205	124 × 206	*	4 × 6전지	단행본
3 × 6판	103 × 182	124 × 206	*	4 × 6전지	문고

여백 여백(margin)은 공란으로 두기도 하고 다른 용도로 활용하기도 한다. 여백의 활용 방법은 상당히 다양하므로 절을 달리하여 논의하기로 한다.

흔히 편집 영역을 '판면'이라 한다. 판면은 양면(두 페이지) 맞쪽으로 이루어진 펼침면으로 디자인되는데 오른쪽 페이지가 홀수쪽이고 왼쪽 페이지가 짝수쪽을 이룬다. 표제지를 제외하고는 모든 페이지가 홀짝수쪽으로 이루어져 있다. 하나의 홀수쪽과 그 뒷면을 '내지(leaf)'라 하고 책으로 묶어졌을 때 홀수쪽의 왼쪽에 오는 앞 장 내지의 뒷면을 '건너편 쪽(facing page)'이라 한다. 홀수쪽을 'recto'라 부르고, 그 건너편 쪽을 'verso'라 부르는 경우도 있다.

시작쪽은 홀수쪽인가 짝수쪽인가

책을 펼쳤을 때 홀수쪽에 먼저 시선이 가는 것은 오랫동안 길들여진 독서 습관이다. 표지가 홀수쪽 위치이므로 시작하는 편(part)이나 장(chapter)은 홀수쪽에 있는 것이 자연스럽다. 그래서 대부분의 학술서적이나 형식을 갖춘 책은 새로운 편이나 장을 시작할 때 왼쪽면이 비더라도 홀수쪽에서 새로이 시작하는 것이 일반적이다.[6] 특별한 경우에 장의 표제를 짝수쪽에 두는 경우가 있다. 장에 에피그라프(epigraph)가 따라오거나 또는 삽화를 넣어서 장식하는 경우도 있다. 그러나 이럴 때라 해도 장 본문의 시작 위치는 홀수쪽인 경우가 많다.

짝수쪽에서 시작하는 책도 있을 수 있다. 조금 덜 형식적인 책이나 페이지 분량이 적은 책의 경우가 그러하고 온라인 문서는 비록 짝수쪽이라 해도 한 페이지를 비우는 것이 부담스럽다. 또한 특별한 디자인 상의 목적 때문에 왼쪽 면에서 시작된 텍스트가 오른쪽 면까지 자연스럽게 이어지게 해야 할 필요가 있을 때도 짝수쪽에서 시작하는 것이 필요하다.

memoir에서는 홀짝수쪽 어디에서든 시작하는 것을 모두 가능하게 하고 있다. 홀짝수쪽에 관련된 memoir의 주요한 명령과 옵션을 정리하면 표 2와 같다.

6. 이것은 mainmatter에 해당하는 말이다. 일반적으로 frontmatter와 backmatter는 `openany`로 조판한다.

표 2. 홀짝수쪽 관련 memoir 명령과 옵션

구분	제어문자열	설명
memoir 옵션	openany	새로운 장을 홀짝수 구분없이 시작한다.
	openright	홀수쪽에서 시작한다.
	openleft	짝수쪽에서 새로운 장을 시작한다.
선언	\openany	openany와 같다.
	\openright	openright와 같다.
	\openleft	openleft와 같다.
페이지 이동 명령	\clearpage	다음 페이지 (라텍 명령)
	\cleardoublepage	다음 홀수 페이지 (라텍 명령)
	\cleartooddpage	다음 홀수 페이지
	\cleartoevenpage	다음 짝수 페이지
	\cleartorecto	다음 홀수 페이지 (강한 명령)
	\cleartoverso	다음 짝수 페이지 (강한 명령)
	\movetooddpage	다음 홀수 페이지 (float clear 않음)
	\movetoevenpage	다음 짝수 페이지 (float clear 않음)

백면 처리

새로운 장을 홀수쪽에서 시작하면 이따금 그 앞 페이지 짝수쪽을 백면으로 비워야 하는 경우가 있다. 잡지 등이라면 이곳에 다른 기사를 실어서 채우겠지만 보통은 그냥 백면으로 둔다.

표준 라텍 클래스는 이 백면에도 면주를 찍어주었기 때문에 불편한 점이 많았다. 이주호가 KTUG 질문 답변 게시판에서 소개한 \cedp라는 명령은 다음과 같이 정의되었는데 \cleardoublepage를 하면서 그 앞의 짝수쪽을 백면으로 만들어주는 것이었다.

```
\newcommand\cedp{\newpage{\pagestyle{empty}\cleardoublepage}}
```

그러나 memoir는 이것을 클래스 자신이 제공하고 있으므로 이와 같은 편법이 필요 없다.

1인치 오프셋

종이 왼쪽 상단에서 가로와 세로 각각 1인치씩 떨어진 곳에 고정점을 가정하여 이 점을 기준으로 조판을 하는 것을 1인치 오프셋이라고 한다. 그러므로 텍에서 다른 아무런 설정 없이 글을 쓰면 이 지점에서부터 조판이 시작된다.

라텍 표준 클래스는 1인치 오프셋을 상정하고 있기는 하나 여백을 1인치로 고정시켜두고 있지는 않다. 클래스에 따라 여백과 편집 영역 길이 설정이 조금씩 차이난다. geometry 패키지는 1인치 오프셋이라는 개념을 라텍에서 제거하도록 해준다. 이 패키지를 로드하면 모든 길이값을 종이의 왼쪽 상단 끝을 기준으로 설정한다.

memoir 역시 이와 마찬가지로 1인치 오프셋을 사용하지 않는다. 그러므로 페이지를 디자인할 때 1인치 오프셋을 의식하지 말고 용지와 판면을 기준으로 구현하면 된다.

표 3. 활자배수조견표(부분)

자수	9pt	10.5pt	12pt
25	79.37mm	92.60mm	105.83mm
30	95.25mm	111.12mm	127.00mm
35	111.12mm	129.64mm	148.16mm
40	127.00mm	148.16mm	169.33mm
45	142.87mm	166.68mm	190.50mm
50	158.75mm	185.20mm	211.66mm

2.2 판면의 구성

판면을 구성하는 방법에 대해서는 수많은 레이아웃 관련 서적과 문헌에서 자세하게 논하고 있다. 이 글은 본격적인 레이아웃에 대해 다루고 있지 않으므로 번거로운 논의는 다른 곳으로 미룬다. 가볍게 일별하고자 한다면 memoir 매뉴얼 [14]에도 간략하나 핵심적인 논의가 있다.

판면을 구성하는 방법은 먼저 판형에 대하여 판면의 크기를 정하고 그 나머지를 여백으로 처리하는 것과 먼저 여백의 크기를 정하고 나머지를 판면으로 설정하는 방법이 있다.

2.2.1 판면의 크기를 먼저 설정하는 경우

판면의 크기를 먼저 정하는 경우에는 행장을 결정하고 이로부터 판면 높이(즉 행수)를 구해야 한다. 이 판면을 페이지의 적절한 위치에 앉히는 것은 여백 설정을 통해서이다. 이것을 '행장–행수' 방법이라고 부르기로 하자. 행수를 통한 판면 높이 설정은 종래 등간격 행간을 의식하고 쓰여온 방법인데 우리는 직접 행수 대신 판면 높이를 행장에서 일정한 비율을 곱하여 얻는 방식으로 접근하기로 한다.

행장 행장이란 행의 길이를 말한다.[7] 행의 길이는 행당 문자수에 의하여 결정된다. 표 3은 [10]에 나오는 것을 축약한 것인데, 카피피팅을 위해서는 이 조견표를 이용한다. 10.5pt 크기의 문자 35자를 넣기로 한 경우 130mm 정도임을 조견표에서 확인할 수 있다.

이 조견표에서 자수 또는 배수라 한 것은 행당 문자수를 가리킨다. 일본어에서는 띄어쓰기가 없고 한 행은 전각문자의 개수에 의하여 결정된다. 그 표를 그대로 가져온 것이므로 띄어쓰기가 있는 우리 문장에는 아무래도 약간의 편차가 생길 수밖에 없다. 게다가 띄어쓰기는 가변길이인 것이다.

여기서 memoir가 채택하고 있는 `\lxvchars`와 `\xlvchars`를 생각해보자. 이 명령들은 현재 기본 폰트에서 65글자와 45글자 길이를 알려주는데, 한 행을 65글자 또는 45글자로 구성하는 서양 조판 관행에 의한 것이다. 그러나 알파벳 길이를 이용하여 카피피팅하는 영문자의 경우와 우리 글자의 경우는 상당히 다를 수밖에 없다. 알파벳 길이가 폰트 디자인에 따라 달라지는 것과는 달리 우리 글자의 경우 사각형 디자인 영역을 가지기 때문이다.

생각건대, 우리 문장의 조판에 있어서 행장을 결정하는 데는 일본식의 배수와 서양식의

7. 여기서 논의하는 행장이란 곧 `\textwidth`이다.

알파벳 길이 및 65자 길이 등을 모두 사용해야 하지 않는가 한다. 왜냐하면 우리 문장은 띄어쓰기가 있는 데다가 문장부호가 모두 영문자 폰트에 의존하고 있으며[8] 그나마 타이포그라퍼들은 폰트 디자이너를 신뢰하지 않아서 마이너스 자간이나 줄인 장평을 흔히 사용하기 때문이다. 예컨대 행장은 '배수'로 결정하면서 글자 박스는 90% 장평으로 줄인다면 약간의 불일치가 발생하지 말라는 법도 없을 것이다.

영문자가 거의 쓰이지 않는 소설 등의 문헌에는 '배수'가 적당하다는 것이 필자의 개인적인 생각이다. 대신 많은 영문자가 등장하는 문헌이라면 배수로 얻어진 값과 알파벳 길이로 얻어진 값을 적절히 조정해야 할 것이다.

우리 문장 한 행은 대학교재 등의 경우 30배 내지 35배 정도로 이루어져 있다. 이보다 길어지면 시선의 이동거리가 멀어져서 눈의 피로감이 커지고 독서의 효율이 낮아질 것이다.

황금비 판면 행장을 얻었다면 그 길이에 1.618을 곱하여 판면 높이를 정할 수 있다. 이렇게 정해지는 판면을 '황금비 판면'이라 한다. 판면의 종횡비를 설정하는 방법은 매우 많지만 오래전부터 황금비 판면에 대해서 많은 논의가 있어 왔다. 아마도 '황금비'라는 말 자체가 묘한 매력을 느끼게 하는 것은 아닐지.

판형의 크기가 황금비가 되는 경우는 예외적이다. 그 까닭은 판형이 용지 크기의 제한을 크게 받기 때문이고 특별한 목적으로 만들어진 것이 아니라면 판형이 황금비일 때 책의 크기가 조금 길어보이기 때문이다. 그러나 판형이 황금비가 아니라도 판면은 황금비로 구성할 수 있다. 황금비 판면이든 어떤 판면이든 이것은 전적으로 디자이너의 선택에 해당한다.

판면의 세로 길이를 황금비로만 구하면 세로의 길이가 상대적으로 길어보일 수밖에 없다. 그래서 이 세로 길이를 면주 영역까지 포함하도록 권장하기도 한다. 그러나 면주 영역에 페이지 번호만이 찍히는 경우라면 이것은 타당하다고 보기 어렵다. 대신 면주에 상당한 양의 정보가 들어가서 적어도 행장의 1/3 이상의 길이를 평균적으로 차지하는 경우라면 면주 영역까지 황금비 높이로 간주하는 방법도 허용된다고 하겠다. 다만 이 경우에는 본문 영역과 면주 영역 사이의 흰 여백이 독서에 미치는 영향을 고려해야 할 것이다.

본문을 10pt로 하고 장평 100%, 자간 0pt일 때 34배 길이를 찾아보면 119.3mm에 해당한다.[9] 여기에 1.618을 곱하면 193.0mm 길이가 되는데, 이로부터 119.3×193.0의 황금비 판면을 얻을 수 있다. 여기에 본문 행간이 1.333이라면 행간 기준치를 12pt로 했을 때 $1.333 \times 12\text{pt} = 15.996\text{pt}$, 약 5.61mm에 해당하는 길이다. 세로 길이를 이 값으로 나눈 값 34.4는 현재의 설정에서 한 페이지에 약 34행이 들어간다는 것을 의미한다. 만약 면주까지 황금비로 한다면 면주 영역 2행을 제외하여 32행이 한 페이지를 이루게 된다.

라텍 표준 book 클래스의 판면 구성을 보면 10pt 문서에서 판면의 가로가 345pt, 세로가 550pt로서 이 비율은 황금비보다 조금 적은 1.594 정도의 값에 해당한다. 상단 면주를 포함하면 1.681이 되어 황금비보다 조금 커진다. 아마도 황금비를 전후한 값으로 이 길이를 잡은 것이 아닌가 싶다.

8. 실은 이것이 한글 문장의 조판을 어렵게 하는 이유 중 하나이다.
9. 1인치, 즉 25.4mm가 72.27pt와 같으므로 1pt의 길이는 약 0.351mm이다.

제안 행장–행수 방법으로 판면을 결정할 때의 지침에 대한 필자의 제안은 다음과 같다.

1. 행장은 문자수 배수를 이용하되 40배를 넘지 않도록 한다.
2. 면주를 제외한 판면의 세로 길이는 행장의 1.53배로 하는 것이 적절하다. 1.5배 (3 : 2)가 아닌 것은 한글 문서의 경우 행간이 길기 때문에 거기서 오는 소실을 방지하기 위하여 1/3을 더 추가한 것이다.
3. 하단 면주는 포함하지 말고 상단 면주를 포함한 판면의 가로 세로 비가 황금비에 가깝도록 면주 위치를 조절한다. 정확하게 황금비가 될 필요는 없지만 면주를 포함한 판면의 세로 길이가 대략 행장의 1.6배를 조금 넘도록 하는 것이 적당하다.
4. 면색인이 길면 판면의 세로 길이를 조금 늘려 잡고 면주를 거의 쓰지 않으면 조금 좁혀 잡는 것이 판면의 안정성에 기여한다.

2.2.2 여백의 크기를 먼저 설정하는 경우

앞서 본 행장–행수 방법과는 달리 먼저 여백을 설정하고 그 나머지를 행장으로 하는 방법도 있다. 라텍에서는 이 방법이 더 직관적이기 때문에 많이 쓰여왔던 것으로 보인다. 여백에는 상단, 하단, 좌, 우의 네 개가 있는데 종래 이 각각을 天, 地, 등, 배로 표기해왔다. 이 글에서는 상, 하, 좌, 우의 표현을 쓰기로 한다.

여백비 서양에서는 여러 가지 여백 처리 방법이 발달해 왔는데 좌–상–우–하의 여백 비를 $1 : 1.2 : 1.2^2 : 1.2^3$ 로 하는 모리스 방식과 $1.5 : 2 : 3 : 4$로 하는 언윈 방식이 있다.

모리스 여백은 자칫하면 판면이 정사각형에 가까워지고 언윈 여백은 우리 책의 일반적인 조판 관행과는 달리 안쪽 위쪽으로 판면이 치우치게 된다. 행간이 좁고 글자가 조밀한 영어권 책에는 적절한 방식일지 모르나 우리 느낌으로는 아무래도 치우친 느낌이 든다.

많은 레이아웃 교재들이 우리 책의 경우 상하 여백을 1 : 1로 할 것을 권장하고 있다 [10]. 행간이 넉넉해서 최하단 한 행의 마지막 행송이 여백의 일부로 지각될 것을 고려하면 1 : 1 정도의 여백비라면 나쁘지 않다는 판단이다. 다만 상단 면주를 쓰는 경우 면주는 여백 영역에 들어가야 한다. 판면이 내려잡히면 불균형한 느낌이 들고 너무 올려잡히면 부담스럽다.

좌우 여백은 1 : 2까지 취하는 경우도 있으나 여백을 살려 여백단 조판을 하지 않는 경우라면 1 : 1.5 정도 (2 : 3)가 적절한 여백비가 아닌가 한다.

면적률 판면의 면적률은 판면면적을 판형면적으로 나눈 값을 가리킨다. 이 값은 일반적인 책의 경우 대체로 50% 내외이면 무난하다. 온라인 문서와 같이 여백이 거의 없는 문서는 60% 가까이 잡으면 된다. 면적률이 50%라 해도 시각적으로는 약 70%가 텍스트 영역인 것으로 인지한다고 알려져 있다 [7]. 라텍 기본 클래스의 면적률은 상당히 낮은 편이다. 행장을 표준적인 65문자에 맞추면서도 용지는 A4 또는 레터를 쓰기 때문에 생긴 일이다. 표준 용지를 사용할 때 적절한 면적률 내지 행장이 얼마여야 하는가에 대해 일률적으로 말할 수는 없는 일이나 A4 크기의 용지라면 40% 정도여야 하지 않을까 싶다.

표 4. memoir의 레이아웃 관련 명령

명령	인자	설정	파라미터
\setstocksize	height, width	2	\stockheight, \stockwidth
\settrimmedsize	height, width, ratio	2/3	\paperheight, \paperwidth
\settrims	top, edge	2	\trimtop, \trimedge
\settypeblocksize	height, width, ratio	2/3	\textheight, \textwidth
\setlrmargins	spine, edge, ratio	(0–1)/3	\spinemargin, \foremargin
\setlrmarginsandblock	spine, edge, ratio	(1–3)/3	
\setulmargins	upper, lower, ratio	(0–1)/3	\uppermargin, \lowermargin
\setulmarginsandblock	upper, lower, ratio	(1–3)/3	
\setcolsepandrule	colsep, thickness	2	\columnsep, \columnseprule
\setheadfoot	headheight, footskip	2	\headheight, \footskip
\setheaderspaces	headdrop, headsep, ratio	(0–1)/3	\headdrop, \headsep
\setmarginnotes	sep, width, push	3	\marginparsep, \marginparwidth, \marginparpush

한글 문서를 작성할 때는 설령 인쇄용지가 A4라 하더라도 190 × 260mm 크기를 염두에 두고 판면편성을 하는 것이 좋으리라고 생각한다. 어찌 되었든 A4 용지를 전제로 여백을 줄이고 행장을 늘리는 것은 좋지 못하다.

2.3 memoir의 판형과 판면 설정

이제 memoir 클래스로 실제 판면 설정을 해보기로 하자. 이 부분은 memhangul-x보다 memoir 자체에서 구현하고 있는 기능을 주로 이용한다. 표 4는 memoir의 레이아웃 관련 명령을 요약한 것이다.

실례를 들어서 이 명령을 이용하여 페이지 레이아웃을 디자인해보자.

2.3.1 용지와 판형

용지를 별도로 설정해야 하는 경우 재단을 위한 트림(trim)마크를 표시하는 것이 좋다. 그러나 PDF를 최종 출력물로 간주하는 경우 용지를 별도로 설정하지 말고 재단된 판형 크기를 용지 크기와 일치시켜두는 것이 편리하다. 각각의 경우로 나누어서 살펴본다.

트림마크 표시 간단히 하기 위해 A4 규격 용지를 사용하자. memoir에 a4paper 옵션이 있지만 우리는 그것을 무시하고 A4 용지의 가로 세로 크기인 210 × 297mm를 직접 이용한다.

```
\setstocksize{297mm}{210mm}
```

이제 판형 크기를 결정해야 한다. 역시 간단히 하기 위해 앞서 지적한 바 있는 190×260mm를 판형 크기로 이용하기로 한다.

```
\settrimmedsize{260mm}{190mm}{*}
```

필요하다면 용지 크기의 비율로 지정할 수도 있다. 이럴 경우에는 예컨대

```
\settrimmedsize{.9\stockheight}{.9\stockwidth}{*}
```

와 같은 방식으로 지정하거나 또는 판형의 세로 길이를 가로 길이의 일정 비율로

```
\settrimmedsize{*}{190mm}{1.368}
```

과 같이 할 수도 있을 것이다.

재단된 판형을 용지 상의 어느 위치에 놓을 것인가? 아무 것도 지정하지 않으면 \trimtop과 \trimedge가 0이기 때문에 판형의 상단 오른쪽 끝이 용지의 상단 오른쪽 끝에 놓인다. 만약 판형의 상단 왼쪽 끝을 용지의 상단 왼쪽 끝에 놓으려면 \trimedge를 용지와 판형의 폭 차이만큼 지정해주면 된다. 이 계산을 라텍이 하도록 한다면 다음과 같다.

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

위의 코드는 \trimedge를 (\stockwidth − \paperwidth) 값으로 설정하는 것이다. 그런 다음에 \trimedge를 \settrims 명령의 두 번째 인자로 주어 판형의 위치를 왼쪽으로 이동시켰다.

만약 판형을 용지의 중앙에 놓고 싶다면 위의 \trimedge 계산을 \trimtop에 대해서도 실행하고 \settrims 명령에는 그 반값을 주어 이동시키면 될 것이다. 다음 코드는 [15, p. 66]에 나오는 것으로 위의 아이디어를 구현한 것이다.

```
\setlength\trimtop{\stockheight}
\addtolength\trimtop{-\paperheight}
\setlength\trimedge{\stockwidth}
\addtolength\trimedge{-\paperwidth}
\settrims{.5\trimtop}{.5\trimedge}
```

트림마크를 표시하게 하려면 memoir 클래스의 showtrims 옵션을 지시하면 된다. 선택할 수 있는 트림마크는 \trimXmarks, \trimLmarks, \trimFrame, \trimNone이 있다. 물론 필요하다면 이 마크를 사용자가 정의하거나 다른 그림으로 교체할 수도 있다 [14]. 우리는 다음 트림마크를 이용하기로 하자.

```
\trimLmarks
```

인쇄물을 만든다면 이와 같은 방법을 써서 트림마크를 표시하는 쪽이 좋다.

용지와 판형이 동일한 경우 최종 출력물이 인쇄물이 아니라 PDF 포맷의 온라인 문서라면 용지에 트림마크 표시를 할 필요가 없고 판형 크기로 PDF가 잘라져야 할 것이다.

이 때는 용지에 대해서 고려할 필요가 없으므로 용지 크기를 판형 크기로 설정하고 재단된 크기를 용지 크기와 일치시키면 된다. 그리고 트림마크에 대해서도 신경쓸 필요가 없으므로 코드는 다음과 같이 간단명료해진다.

```
\setstocksize{260mm}{190mm}
\settrimmedsize{\stockheight}{\stockwidth}{*}
```

용지라는 개념은 라텍 표준 클래스에는 없는 memoir 특유의 것이다.

2.3.2 편집 영역의 설정

편집 영역(판면)을 설정하는 데는 여러 가지 방법이 있다. 코딩 전에 행장과 면주, 그리고 여백은 미리 결정되어 있어야 한다. 이 길이들을 미리 그리드에 그려두고 코딩을 시작하면 편리하다.

행장을 미리 정하고 여백을 설정하는 방법 memhangul-x에는 `\setxxxlength`라는 재미있는 명령이 있다. 이것은 다음과 같이 쓰인다.

```
\setxxxlength{xxxvi}{36}
한글 36문자의 길이는 \underline{\the\xxxvilength}.
```

첫 번째 인자는 이 명령의 수행 결과를 담고 있는 제어 문자열의 이름을 지어주기 위한 것이고 두 번째 인자는 숫자이다. 위의 경우 한글 36글자를 연이어 썼을 때의 길이가 `\xxxvilength`라는 제어 문자열에 들어가게 된다. 폰트 크기나 장평을 바꾸면 이 값이 변할 것이다.

이 매크로를 이용하여 행장을 정할 수 있다. 그런 다음 여기에 1.53을 곱하여 편집 영역의 높이를 얻는다.[10]

```
\setxxxlength{xxxvi}{36}
\settypeblocksize{*}{\xxxvilength}{1.53}
```

물론 다른 방법으로 편집 영역을 설정하는 것도 얼마든지 가능하다. 편집 영역의 가로 세로 길이를 미리 정하여 두었다면 예컨대

```
\settypeblocksize{200mm}{120mm}{*}
```

와 같은 방식으로 값을 지정하여도 될 것이고, 원한다면 판형의 가로 세로 길이의 배수 형식으로 지정하여도 될 것이다.

그 다음으로 좌우 여백비를 설정하여야 한다. 여백 안에 문단을 별도로 두지 않고 내측 여백(`spinemargin`)과 외측 여백(`foremargin`)을 2 : 3 비율로 한다면 다음과 같이 비율만을 정해주는 것이 좋다. 필요하다면 내측 여백이나 외측 여백 값을 직접 지시할 수 있다.[11]

```
\setlrmargins{*}{*}{1.5}
```

상하 여백도 같은 방식으로 설정할 수 있다. 상하 여백비를 다음과 같이 1 : 1로 설정한다.

```
\setulmargins{*}{*}{1.0}
```

지금까지의 설정으로 만들어지는 문서의 레이아웃이 어떠할지 상상해볼 수 있겠는가? memoir 클래스가 `\checkandfixthelayout` 명령이 주어졌을 때 출력하는 내용은 다음과 같다.[12] 여기서 예시한 예제는 면적률이 약 42%로서 여백이 조금 넉넉한 편에 속한다.

10. 왜 1.53을 예로 들고 있는가는 8쪽 '제안'을 보라.
11. 이 둘을 모두 지정하면 memoir가 에러를 낼 것이다. 왜냐하면 `textwidth`가 이미 정해져 있고 `paperwidth`도 정해져 있는데 `spinemargin`과 `foremargin`이 모두 절대값으로 주어질 수가 없기 때문이다. 어느 하나만 정해주어야 다른 값을 `paperwidth`와 `textwidth`에 대하여 계산할 수 있다.
12. 아직까지 `columnsep`, `headheight`, `marginparwidth` 등은 지정하지 않았으므로 기본값이 사용되었다.

```
**************************************************
Stock height and width: 845.04684pt by 597.50787pt
Top and edge trims: 52.63759pt and 28.45276pt
Page height and width: 739.77165pt by 540.60236pt
Text height and width: 505.87323pt by 332.0pt
Spine and edge margins: 83.28247pt and 124.92633pt
Upper and lower margins: 115.60059pt and 118.29784pt
Headheight and headsep: 12.0pt and 18.06749pt
Footskip: 25.29494pt
Columnsep and columnseprule: 10.0pt and 0.0pt
Marginparsep and marginparwidth: 7.0pt and 128.0pt
**************************************************
```

여백을 먼저 설정하는 방법 이번에는 반대로 여백을 설정하고 행장은 자동으로 정해지도록 하는 방식으로 접근해보자. 여백을 중시하는 디자이너들은 이런 방법을 선호하기도 한다. 이 경우 여백값이 미리 결정되어 있어야 한다. 그런데 조화로운 판면을 구성하는 데 있어서 미리 여백값을 결정해두기란 쉬운 일이 아니다. 그래서 일반적으로는 앞서 소개한 행장을 먼저 결정하는 방법을 권장한다. 그러나 앞으로 논의할 변이단 조판 등에서는 여백값을 미리 결정해두는 방법을 쓰는 것이 좋다.

여백값을 지정하는 방법으로 접근할 경우에도 행장을 미리 정해두고 할 수 있다. 예컨대 [10, pp. 83f]에 소개하고 있는 여백값 정하기 방법은 미리 행장을 정해둔 다음 여백의 값을 차례로 계산하고 있다. 우리는 면적률을 통하여 행장이 맨마지막에 자동으로 결정되는 방법으로 이 문제에 접근해보고자 한다. 어떤 방법이든 디자이너의 의도가 중요할 것이다.

면적률이 50%가 되도록 여백을 정해보자. 판형은 앞서 도입한 190×260mm 라고 하자.[13] 좌우 여백의 비가 $2:3$이 되게 하고 상하 여백은 $1:1$이 되게 하려 한다.

판형의 가로 및 세로 길이를 각각 W와 H라 하고, 판면의 가로 및 세로 길이를 각각 w와 h라 하자. 그리고 내측 여백 l에 대한 외측 여백의 비(fore/spine)를 r이라 하고, 상단 여백 u에 대한 하단 여백의 비(lower/upper)를 p라 하면 다음 관계식이 성립한다.

$$(1+r)l = W - w \qquad (1+p)u = H - h$$

판면종횡비 h/w를 R이라 하고 면적률(판형 면적에 대한 판면 면적의 비)을 F라 하면

$$F(HW) = hw = Rw^2 = h^2/R$$

이므로 다음과 같이 내측 여백과 상단 여백에 관한 식을 얻을 수 있다.

$$l = \frac{W - \sqrt{FHW/R}}{1+r} \qquad u = \frac{H - \sqrt{RFHW}}{1+p}$$

위의 식에 $F = 0.5$, $R = 1.53$, $r = 1.5$, $p = 1$, $W = 190$, $H = 260$을 각각 대입하면 내측 여백 $l = 25.2$, 상단 여백 $u = 32.75$, 그리고 외측 여백 $lr = 37.8$을 얻는다.

이제 이것을 다음과 같이 구현하자. `\settypeblocksize` 명령을 쓰지 않고 다음 명령으로 여백을 통해 편집 영역의 크기가 자동으로 결정되도록 한다.

13. 190×260mm 크기는 흔히 '사륙배판'이라 부르는 크기와 유사하다.

```
\setlrmarginsandblock{25.2mm}{37.8mm}{*}
\setulmarginsandblock{32.75mm}{*}{1.0}
```

이 결과는 판면의 면적률은 적당하지만 행장이 10pt 본문 크기의 39배 가량 되어서 경우에 따라 조금 긴 듯한 느낌을 줄 수 있다. 이 정도 길이의 행장이 나오는 판형이면 자간도 너무 줄이지 말고 본문 10.5pt 내지 11pt를 쓰는 것이 좋으리라 생각한다.

2.3.3 면주 영역과 여백 문단의 설정

면주 영역을 설정하여 위치를 잡는 것은 생각보다 까다로운 점이 있다. 면주의 폰트 크기나 폰트 패밀리에 따라서도 고려해야 할 점이 있다. 이것은 페이지 스타일이 결정되어야 제대로 다룰 수 있을 것이다. 여기에서는 `headings` 페이지 스타일이 사용된 경우를 상정해서 상단 면주만 있는 것으로 하고 이 경우 면주 영역을 설정하는 방법만을 간단히 정리하기로 한다.

면주[14] 영역을 설정하려면 `\headheight`와 `\footskip` 값을 지정한다. `\headheight`는 상단 면주의 높이를 가리키는 길이이고 `\footskip`은 `\textheight`의 하단 끝에서 하단 면주의 끝까지 길이를 가리키는 것이다. 앞서 설정한 레이아웃에서 이 값은 각각 12pt와 18pt로 되어 있는데 그대로 사용하는 것이 문제가 없다면 그냥 두어도 된다. 이보다 더 실용적인 명령으로 `\setheaderspaces`가 있는데 이 명령은 `\headdrop`과 `\headsep`을 설정한다.

여백 안에 위치하는 문단의 폭은 `\marginparwidth` 값으로 설정하고, 여백 안의 위치는 `\marginparsep`과 `\marginparpush`로 설정한다. 여백 안에 문단을 쓰지 않고 여백을 다만 비우는 공간으로만 사용하려 한다면 이러한 값들을 별도로 설정하지 않아도 된다.

아래 예제는 `\headheight`를 16pt로, `\footskip`을 30pt로 한 다음 `\headsep`을 18pt로 하여 `\headdrop`은 자동으로 계산하게 한 것이다. 그리고 `\marginparsep`을 10pt로, `\marginparwidth`를 100pt로, `\marginparpush`를 10pt로 지정한다.

```
\setheadfoot{16pt}{30pt}
\setheaderspaces{*}{18pt}{*}
\setmarginnotes{10pt}{100pt}{10pt}
```

여백 안에 위치하는 문단의 폭은 경우에 따라 여백 크기보다 클 수도 있다. 그러나 그럴 경우 디자이너는 자신이 무슨 일을 하고 있는지 알고 있어야 한다.

2.3.4 check and fix

memoir는 레이아웃 설정이 끝나면 마지막에 다음 명령을 반드시 실행하도록 요구하고 있다.

```
\checkandfixthelayout
```

레이아웃 설정값들을 검토하여 불일치가 있으면 에러 메시지를 보여주고 그렇지 않으면 텍의 길이들을 설정하여 효과가 발생하게 한다.

14. 면주는 folio의 역어로 채택되었으나 여기서 면주라고 부르는 것은 면번호와 헤딩 텍스트를 모두 가리킨다.

2.4 fapapersize를 이용한 판면 설정

앞에서 길게 논의한 판면 결정이 이루어졌다면 oblivoir의 부속 패키지인 fapapersize를 이용하여 간단히 판면을 결정할 수 있다. 실제 fapapersize가 하는 일은 위에서 논의한 memoir의 판면 결정 명령을 주어진 값으로 설정해주는 것이다.

fapapersize의 용지 크기 옵션은 사륙배판(dbl4x6), 국판(mum), 신국판(newmum)이 있다.

```
\documentclass{oblivoir}
\usepackage[dbl4x6]{fapapersize}
```

이렇게 하면 사륙배판 용지 크기 190 × 260mm에 적절하게 여백이 들어간 판면을 얻는다.

fapapersize를 아무런 옵션 없이 쓰면 아무 일도 하지 않지만, \usefapapersize 명령으로 편집 영역의 크기와 여백을 조절할 수 있다. 이 명령에서 제공하는 여섯 파라미터는 차례대로 \paperwidth, \paperheight, \leftmargin, \rightmargin, \uppermargin, 그리고 \lowermargin을 의미한다. 단, 옵션도 주고 이 명령도 쓰는 경우는 옵션이 우선한다.

```
\usepackage{fapapersize}
\usefapapersize{210mm,297mm,30mm,*,30mm,32mm}
```

특히 \rightmargin이 \leftmargin과 동일할 때에는 네 번째 파라미터를 간단히 별표(*)로 줄 수 있다. 마찬가지로 \lowermargin이 \uppermargin과 동일할 때에도 여섯 번째 파라미터에 별표를 표시하면 된다.

별도로 판형 크기를 설정하고 싶지 않고 기본값 또는 문서 옵션에서 주어진 크기를 그대로 쓰고 여백만 설정하려 할 때는 다음과 같이 첫 번째와 두 번째 파라미터에도 별표를 쓸 수 있다.

```
\usepackage{fapapersize}
\usefapapersize{*,*,30mm,*,30mm,*}
```

fapapersize의 옵션으로 stock이 주어지면 임의의 용지 크기를 설정할 수 있다. 임의 용지 크기는 \usefastocksize 명령으로 설정하는데 성격상 \usefapapersize 명령보다 먼저 쓰여야 한다.

특별히 \setheadfoot, \setheaderspaces, \setmarginnotes와 같은 명령을 써야 하는 상황이라면 \usepackage{fapapersize}와 \usefapapersize 명령 사이에 두어야 효과가 발생한다.

3 글꼴 : 글꼴가족

최근 가장 많은 변화가 있는 부분이 글꼴 사용에 관련된 것이다. XETEX을 통하여 오픈타입, 트루타입 글꼴을 자유롭게 활용하게 됨으로써 종래 텍 시스템에서 한글을 구현하기 위해 글꼴 문제를 해결해야 했던 부담이 크게 줄었다.

oblivoir에서의 글꼴 사용은 기본적으로 fontspec 패키지를 이용하는 것으로서 XETEX-k*o*의 것을 그대로 따른다. 폰트 속성을 이용하는 복잡한 논의는 이 글의 범위를 벗어나는 것이므로 여기서는 한글 글꼴의 선택에 대한 것만 다루기로 한다.

3.1 서체와 폰트족

서양 글자를 표현하는 폰트는 몇 개의 폰트가 하나의 가족(family)을 이룬다. 예컨대 upright, *italic*, **bold**, ***bold-italic*** 등을 한 묶음으로 하여 하나의 폰트족이 되는 것이다. 그런 다음에 rmfamily, sffamily, `ttfamily`라는 세 개의 폰트족으로 문서의 기본 폰트 그룹을 이루도록 한 것이 라텍의 폰트 사용 방식이다.

우리 글자에는 폰트족이라 할 만한 것이 없다. 명조군에서는 세명, 중명, 태명, 견명의 구별이 하나의 폰트족을 이룬다고 할 수 있겠고, 윤체의 경우처럼 110, 120 등 두께를 나타내는 일련의 폰트들을 묶어서 하나의 가족으로 처리할 수 있을 듯하다. 우리 글자에 이탤릭은 존재하지 않는다.

본문 글꼴로는 본문체가 쓰이는데 예전부터 '명조'라 불렀다. 최근에는 '바탕'이라는 이름도 많이 쓰이는 것 같지만 여전히 명조라는 명칭이 대세인 듯하다. 원래 명조라는 것은 한자 서체를 가리키는 것이었으나 이제는 본문체에 상당하는 새로운 의미를 얻었다고 생각된다. 세리프가 없는 이를테면 영문자의 산세리프에 해당하는 글꼴을 '고딕'체라고 부른다. 이밖에 궁서 내지 해서체가 있다고는 하나 본문에서 특별한 목적 이외에 쓰기에는 적절하지 않아 보이는 붓글씨체이다.

폰트의 사용과 관련해서 한 마디 하고 싶은 것은 제목 글꼴의 과도한 사용이다. 우리나라 글꼴은 이상하게도 본문용 글꼴은 거의 개발되지 않고 제목 글꼴만 수없이 생산·유포되고 있다. 포스터나 브로슈어에서나 쓰일 법한 것이 TV의 광고와 자막에도 등장하고 심지어 일반 서물에까지 이런 글꼴이 무분별하게 쓰이고 있다. 그것이 디자이너의 선택이라면 할 말은 없지만 일관성과 친숙함이 요구되는 학술 서적의 경우는 본문용 서체가 제목 서체보다 훨씬 중요하다.

어느 논문에서 본문 서체로 윤명조가 가장 선호된다는 연구결과를 보여준 바가 있다 [6]. 이것은 무엇보다도 서체의 아름다움과 익숙함에서 비롯되는 것이라고 생각되지만 워드 프로세싱 환경에서는 윤명조보다 오히려 윈도 기본 글꼴의 친숙함이 더 크지 않은가 싶다. 윈도 기본 글꼴은 PDF로 만들었을 때 획이 가늘고 자면이 너무 커서 불균형해보이는 단점이 있음에도 불구하고 윈도라는 운영체제 덕분에 가장 널리 쓰이는 '리포트용 서체'가 되었다.

3.2 글꼴 영역

XeTeX-k*o*는 문자의 식자 범위를 크게 세 영역으로 구분한다. 각각은 라틴, 한글, 한자 영역이라 불리우는데 종래 T1 인코딩의 범위에 해당하는 영문자와 유럽문자들이 라틴 영역에 들어가고, 유니코드 한글 완성 문자 영역이 한글 영역에, 그리고 그밖의 문자들이 한자 영역에 들어간다. 원칙적으로 이 세 영역을 다른 글꼴로 식자할 수 있게 한 것이다. 다만 한자 영역은 편의상 별도로 지정하지 않으면 한글 글꼴과 동일한 글꼴로 식자한다.[15] 이렇게 복잡한 영역 구분이 필요한 이유는 근본적으로 글꼴의 속성 때문이다. 즉 한글 글꼴에 들어 있는 라틴 문자

15. LuaTeX-k*o*는 이와는 다른 방식으로 세 영역을 식자한다. 마치 레이어를 쌓아가듯이 맨 위쪽부터 라틴–한글–한자 순서로 글꼴을 쌓아놓은 다음 맨 위쪽의 비어있는 글자를 그 아래 레이어에서 찾는 방식이다. 따라서 XeTeX-k*o*와 LuaTeX-k*o*의 글꼴 지정명령은 유사해도 결과가 다르게 나타날 수 있다.

표 5. 글꼴 영역과 글꼴군

	roman (main)	sans serif (sans)	typewritter (mono)
라틴	\setmainfont	\setsansfont	\setmonofont
한글	\setmainhangulfont	\setsanshangulfont	\setmonohangulfont
기타문자	\setmainhanjafont	\setsanshanjafont	\setmonohanjafont

를 문서에서 그대로 쓰기에는 불만족스러운 경우가 많고 한글 글꼴이라 해도 한자가 결락된 경우도 많아서 하나의 폰트만으로 한글 문헌이 요구하는 모든 문자를 충실하게 식자할 수 없다는 결론에 도달했기 때문이다.

표 5에 글꼴 지정 명령을 요약해두었다. 이 명령들은 모두 fontspec 문법에 따라 쓸 수 있다. 예컨대 다음과 같이 옵션을 주는 것이 가능하다.[16]

```
\setmainhangulfont[Ligatures=TeX]{HCR Batang LVT}
```

이런 식의 명령이 마련된 이유는 라텍의 NFSS 명령과 대응하게 하기 위함이다. 즉 main, sans, mono 폰트들은 각각 \rmfamily, \sffamily, \ttfamily 명령에 대응한다.

글꼴 관련 명령

XeTeX-k*o*가 마련하고 있는 글꼴 관련 명령을 요약한다.

- fontspec 패키지의 \newfontfamily와 \newfontface 명령을 해당 영역에서 사용할 수 있도록 \newhangulfontfamily, \newhanjafontfamily, \newhangulfontface, \newhanjafontface 명령을 마련하고 있다.
- \fontspec 명령을 해당 영역에서 쓸 수 있는 \hangulfontspec과 \hanjafontspec 명령이 있다. 일시적으로 특정 폰트로 식자할 때 사용한다. 각각 \adhochangulfont 및 \adhochanjafont 명령과 같다.

기호 글자의 영역 할당

사용자가 다음 명령으로 라틴 문자에는 'Times New Roman'을, 한글에는 '함초롬 바탕'을 기본 글꼴로 선택했다고 가정해보자.

```
\setmainfont[Mapping=tex-text]{Times New Roman}
\setmainhangulfont{HCR Batang LVT}
```

그런데 괄호 글자, 구두점 등 주로 문장부호에 해당하는 기호 글자는 어떤 영역의 글꼴로 식자될 것인가? \xetexkofontregime 명령은 기호 글자들에 영역을 할당해 주는 매우 중요한 명령이다. 표 6은 이 명령의 인자와 옵션에 오는 값들을 요약한 것이다.

예를 들어 다음 명령은 따옴표는 한자 영역 폰트에서, 마침표는 한글 영역 폰트에서, 그 밖의 기호 글자들은 라틴 영역 폰트에서 가져온 것을 식자하라는 명령이다.

```
\xetexkofontregime[quotes=hanja,puncts=hangul]{latin}
```

16. 이 옵션에 관한 자세한 사항은 fontspec 매뉴얼 [13]을 참고하라.

표 6. fontregime의 값

선택인자	영향을 받는 문자	영역인자
alphs	라틴 알파벳	latin
nums	라틴 텍스트 숫자	hangul
parens	라틴 괄호	hanja
quotes	따옴표	prevfont (직전 폰트)
colons	콜론, 세미콜론, em-dash, en-dash	
hyphens	하이픈	
puncts	마침표, 물음표, 느낌표, 쉼표	
cjksymbols	CJK 구두점, 괄호, 상징기호	

3.3 oblivoir의 관점

oblivoir는 한글 글꼴 설정을 위해 \setkormainfont, \setkorsasfont, \setkormonofont 명령을 제공한다. 이 명령군의 목적은 한글 및 한자 영역 글꼴을 그 이름만으로 간편하게 지정하자는 것이다. 이에 대해서는 [3]에 자세하게 소개되어 있다.

이 유형의 명령군은 두 가지 방식으로 사용할 수 있다. 옵션 인자를 각괄호로 쓰는 방식과 괄호로 쓰는 방식이다. 전자는 기본적으로 XeTeX-k*o*의 것과 동일하지만 한글 및 한자 글꼴 설정을 잇대어 쓰게 한다. 예를 들면

```
\setkormainfont[BoldFont=나눔고딕,Mapping=tex-text]%
                {HCR Batang LVT}[BoldFont=돋움]{바탕}
```

위와 같이 씀으로써 한글 글꼴 'HCR Batang LVT'에 해당하는 첫 번째 옵션 '[BoldFont=나눔고딕,Mapping=tex-text]'과 한자 글꼴 '바탕'에 해당하는 두 번째 옵션 '[BoldFont=돋움]'을 따로 줄 수 있다.

```
\setkormainfont(나눔고딕)(*){나눔명조}(한양해서)(*){바탕}
```

위와 같은 용법은 글꼴 이름만을 나열함으로써 간단히 문서의 기본 글꼴을 지정해주는 방식이다. 불필요한 대부분의 옵션을 생략할 수 있게 되어 있다.

3.4 글꼴 이름

XeTeX은 글꼴을 두 가지 방식으로 찾는다. 하나는 텍 시스템의 응용 프로그램들이 파일을 찾을 때 공통으로 사용하는 'kpsearch' 라이브러리를 통해 '글꼴 파일 이름'을 찾는 방식이고 다른 하나는 운영체제에 등록된 '글꼴 이름'으로 호출하는 방식이다.

예컨대 'HANBatang-LVT.ttf'라는 글꼴 파일 이름을 가진 폰트를 운영체제에 등록하면 '함초롬바탕 LVT' 또는 'HCR Batang LVT'라는 글꼴 이름으로 참조할 수 있다.[17]

17. 오래된 한글 글꼴 중에는 이 글꼴 이름이 잘 등록되어 있지 않아서 사용상 장애를 일으키는 경우가 있다. '한겨레결체'가 그런 경우이다. 폰트의 내부 정보인 글꼴 이름 테이블을 수정해주면 문제가 없어진다.

4 글꼴 : 크기, 자간, 장평

4.1 본문 글꼴의 크기

영문자의 폰트는 그 폭이 글자마다 다르고 많은 리거처가 있지만 국한문의 폰트 자체는 네모꼴로 이루어진다. 그러나 실제 조판에서는 대부분의 문장부호가 정사각형 안에 갇히는 꼴이 아니라 가변폭 글자들로 이루어져 있기 때문에 한글은 전부 글자폭이 같다는 것은 올바른 말이 아니다. 문장부호도 한글 문장의 일부를 구성하는 데다가 띄어쓰기의 스페이스는 분명히 가변폭이기 때문이다.

최근 들어 본문 글꼴이 커지는 경향을 보여서 본문 11포인트[18] 서적도 심심찮게 볼 수 있다. 그 이유로 종이를 특별히 절약해야 할 유인이 적어진 점과 독서인구의 시력 감퇴를 드는 경우가 있는데 개인적으로는 이것이 그다지 필요하다고 생각하지 않는다. 예전 활판 시절에는 구비하고 있는 활자의 호수가 포인트로 말하자면 9포인트와 10.5포인트가 주류였기 때문에 큰 글자로 박으면 10.5포인트이고 9포인트 조판이 대세였다. 그러나 컴퓨터 조판이 시작되면서 활자의 호수 또는 급수를 비교적 자유롭게 조절하게 된 지금은 이와 같은 활자 크기의 제약이 사실상 없다고 해야 하겠다. 큰 크기의 폰트에 반대하는 이유는 독서의 효율 때문이다. 11포인트까지 폰트를 키우면 한 단어가 차지하는 폭이 너무 넓어져서 단어가 직관적으로 읽어지지 아니하고 한 글자씩 떼어 읽어야 이해가 된다. 우리가 책을 읽을 때는 글자 단위가 아니라 단어 단위 또는 어절 단위로 읽는다는 점을 고려해보면 10포인트 본문이야말로 가장 적절한 크기가 아닌가 한다.

같은 10포인트라 해도 폰트마다 실제 크기는 조금씩 다르다. 가령 한양 계열의 폰트는 10포인트 활자가 제법 크지만 은글꼴은 상대적으로 조금 작다. 산돌 명조 계열이 필자가 알고 있는 한 자면이 가장 작은 글꼴이 아닌가 한다. 필자는 산돌 명조 정도의 10포인트가 사륙배판(표 1 참조) 크기의 서물에서 가장 좋다고 생각하고 있지만 이것은 어디까지나 주관적인 판단일 뿐이다.

4.2 자간

자간이란 글자 사이의 간격을 의미한다. 서양 문서에서 자간은 거의 문제가 되지 않는데 그 이유는 글자마다 폭이 다 다를 뿐 아니라 글자와 글자가 이어질 때 그 간격을 얼마로 할 것이냐의 문제가 '폰트 디자인' 의 문제로 취급되기 때문이다. 다양한 종류의 리거처[19]를 가진 폰트의 경우에는 자간이 무의미할 수밖에 없을 것이다. '마이크로소프트 워드' 와 같은 워드프로세서는 폰트 디자인 상의 자간을 무시하고 사용자가 임의로 자간을 변경할 수 있도록 해두고 있는데 사실은 이것이 워드로 작성된 문서가 엉성하게 보이게 하는 주범이다.

우리나라나 일본은 한 글자 단위로 식자한다. 그 결과 자간의 문제가 문서 전체의 판면

18. 포인트라는 단위는 일반적으로 포스트스크립트에서 사용하는 1/72인치를 일컫는다. 텍에서 사용하는 포인트 단위는 1/72.27인치인데 pt로 표시한다. 대신 포스트스크립트의 1/72인치에 해당하는 포인트는 bp로 표시한다.
19. 합자. 영어권 등에서 특별한 두 글자가 이어질 때 각각 별도의 활자로 식자하지 않고 두 글자가 이어진 하나의 활자를 사용하는 것을 말한다. 예를 들면 'fi' 가 아니라 'fi' 로 식자하는 것이 대표적이다.

인상을 결정하는 중요한 요소가 된다.

한글 문서 조판에서 '마이너스 자간'은 상당한 이론적 근거와 경험적 축적이 있다. 주로 가로쓰기에서 글자를 세로로 조금 길게 보이도록 장평을 줄여주고 그 대신 글자 사이를 당기는 방식으로 조판이 시도되기 시작한 것은 멀리 잡아도 사진식자가 시작된 80년대를 넘어가지 않는다. 마이너스 자간이 가독성을 높인다는 주장도 있다. 이제는 다음과 같은 견해가 거의 상식이 되었다 [8, p. 202].

> 지나치게 좁은 글자 사이나 낱말 사이 띄우기는 서로 겹쳐보여 글읽기를 방해하는 요인이 되겠지만 마찬가지로 지나치게 넓은 글자 사이와 낱말 사이 띄우기는 독서시 자연스러운 눈의 운동을 방해한다. 글자 사이와 낱말 사이의 공간은 반드시 글자의 넓이와 비례해서 조정되어야 한다. 예를 들어 '빼'와 '기'의 폭은 다르게 조정되어야 할 것이다. 컴퓨터에서 이루어진 폰트 디자인은 1000×1000의 수치에서 작업되는데 '빼'가 1000에 들어가려면 상대적으로 '기'의 폭은 좁아질 수밖에 없는 것이다. 평균적인 자폭은 800 수준이므로 자간 없이 조판할 경우 글자와 글자 사이는 200씩 벌어지게 되어 자간이 매우 벙벙해보이게 된다. 그러므로 한글의 자간은 가능한 한 자간을 좁히는 것이 낱말인지를 쉽게 해준다는 원리가 있다.

이 글에서 생각해볼 것은 "가능한 한 자간을 좁히는 것이" 낱말 인지를 쉽게 해준다고 하였는데 사실 이것은 글자의 모음이 어떤 위치에 있느냐에 따라 그 다음 자간이 달라져야 한다고 말할 수 있다. 특히 'ㅏ'와 같은 모음은 다음 글자의 모음이 가로획('ㅡ' 등)일 때 마이너스 자간을 주는 경우 거의 예외없이 모음끼리 이어붙는다.

평균적인 자폭을 기준으로 마이너스 자간을 주장하는 것은 때에 따라 위험하다. 우리가 글자 자체를 읽는 것이 아니라 단어 전체의 윤곽을 통해서 글을 인지한다고 할 때 돌출획과 가로획이 이어붙는 상황이 눈에 익숙해질 것 같지 않다. 즉 마이너스 자간은 오히려 독서를 방해하는 요소가 될 수 있다는 것이 필자의 생각이다. 마이너스 자간의 남용으로 책을 읽으면서 실제로 고통을 겪은 경험이 필자에게는 있기 때문에 위의 주장이 일견 타당함을 인정하지만 예외없이 동조하기 어렵다. 다만 ko.TEX이 기본 글꼴로 채택한 은글꼴 트루타입은 바탕체의 경우 자면이 가득차지 않는 폰트이기 때문에 약간의 마이너스 자간이 판면에 좋은 효과를 가져오는 것으로 판단한다.

memhangul-x는 자간 기본값을 0pt로 하고 있다. 그 이유는 이 스타일이 은글꼴만을 기본 글꼴로 채택하는 상황을 염두에 두고 있지 않기 때문이다. 윈도 기본 글꼴과 거의 동일한 한양 글꼴과 같은 경우 ──폰트의 품위는 차치하고── 자면 가득차는 큰 형태의 글꼴이기 때문에 장평을 줄이지 않고 마이너스 자간을 쓰면 인접해 오는 가로획 모음이나 돌출획 모음이 엉켜서 오히려 가독성을 떨어뜨릴 수 있다.

즉, 자간은 폰트 디자인과 밀접하게 결부되어 있는 것이라고 생각한다. 그 크기가 얼마가 되어야 하는가는 전적으로 디자이너의 판단에 달린 것이지만 아주 작은 정도가 아니면 마이너스 자간을 남용하는 것은 권장하기 어렵다. 마이너스 자간을 굳이 사용하겠다면 장평을 조금

줄인 다음 적용하라. 만약 최선의 값이 얼마인지 확신할 수 없다면 0pt를 기준으로 판면의 색조(명암도)와 조판 결과를 보고 판단하는 것이 좋지 않을까 한다. 그리고 이것은 전적으로 어떤 폰트를 사용했느냐에 절대적으로 좌우된다.

자간은 문서의 내용과도 무관하지 않다. 자간이 좁아지면 날렵해지고 모던한 느낌을 주지만 0pt 내외의 자간은 장중하고 무게있는 문서에 적당하다. 디자이너 ──저자 자신이 자간과 행간 문제를 다룰 때는 디자이너이기도 하다── 는 이런 여러 가지 점을 고려하여 자신의 문서에 적절한 자간을 선택하여야 한다.

4.3 장평

장평이란 한 글자 박스의 세로 대 가로 비를 말한다. 예컨대 글자 박스의 가로(width)가 9pt이고 세로(totalheight = height + depth)가 10pt라면 장평을 90%라고 한다. 대부분의 한글 글꼴은 주어진 디자인 크기(em)를 가로와 세로로 하는 정사각형 박스에 디자인된다. 장평을 줄이거나 늘린다는 것은 이 세로 대 가로의 비율을 변경하는 것이다.

흔히 알려진 바에 따르면 가로쓰기의 경우에는 세로가 조금 긴 것이 가독성을 높이고 세로쓰기의 경우에는 가로가 조금 긴 것이 읽기 쉽다고 한다.

폰트마다 조금씩 디자인상의 편차가 있다. 예컨대 한양 바탕체는 디자인 박스 안에서 좌우 공간의 여분이 상하의 여분보다 조금 길어서 글자 자체가 아주 조금 세로가 긴 모양을 하고 있다. 반면 신문명조와 같은 글꼴은 디자인이 동일한 크기의 정사각형 안에서 이루어졌다 해도 상하 여분이 좌우 여분보다 크기 때문에 가로가 길어 평평한 느낌을 주는 폰트이다. 그러므로 한양 바탕 글꼴에 장평을 줄이면 세로가 길쭉한 날렵한 모양이 되지만 신문명조의 장평을 줄이면 그와 같은 효과를 얻지 못한다. 신문명조는 그 이름이 말하는 바와 같이 ──오늘날 세로쓰기를 거의 하고 있지 않지만── 세로쓰기 신문 제작 시절의 폰트를 흉내낸 것이다. 즉 세로쓰기에 적당한 서체를 바탕으로 해서 만들어진 것이다.

앞서 언급한 바와 같이 한양 바탕(즉, 윈도 바탕)은 좌우에 아주 미세한 여분이 있기 때문에 이것을 트리밍해주는 효과를 아주 작은 마이너스 자간으로 얻을 수 있다. 나아가 장평을 조금 줄이면 그러한 효과를 좀 더 확연하게 얻게 된다.

주로 젊은 층을 대상으로 한 가벼운 읽을거리의 경우 조금 장평을 줄이고 마이너스 자간을 적용하는 것이 한 가지 디자인 상의 선택이 될 수 있을 것이다.

장평을 줄이려 할 적에 고려해야 될 요소가 한 가지 있다. 트루타입이나 포스트스크립트 한글 글꼴을 이용하면 장평을 조정하는 것은 쉬운 일이지만 기계적 장평 줄임은 세로획이 빈약해지는 결과를 가져온다. 그리고 가끔 'ㅇ'과 같은 글자가 왜곡되는 수도 있다. 이런 점을 유념하여 장평을 선택하여야 할 것이다.

레이아웃 프로그램이나 워드 프로세서는 장평을 조절하는 것이 자유롭다. 라텍에서는 그다지 쉽지 않았으나 최근 XeTeX의 도움으로 장평 조절이 매우 수월하게 되었다.

그러나 다시 강조하거니와 줄인 장평은 특별한 디자인적 요구가 있을 때만 사용하는 것이 좋다. 일반적인 보고서 등에서 줄인 장평은 경우에 따라 품위의 저하를 가져올 수 있다. 당연한 일이지만 memhangul-x의 장평 기본값은 100%이다.

5 간격 : 어간, 행간

5.1 어간

단어 간격을 어간이라고 한다. 자간이 일반적으로 고정 간격인 데 비해 어간은 공백의 상하한을 지정하는 가변 길이 값이다. 그렇게 하는 이유는 양끝맞추기(justification) 때문인데 행의 양끝을 맞추기 위해서 단어 사이를 늘리거나 줄인다. 이 허용폭의 상하한이 너무 차이가 많이 나면 이따금 너무 넓은 어간으로 인해 페이지 색조의 흰색 반점이 생겨나게 되고 너무 좁혀두면 양끝맞추기가 어려워진다.[20] Peter Wilson은 memoir 매뉴얼에서 다른 타이포그라피 책을 인용하여 적절한 영문자 자간은 'i' 의 폭에 해당한다는 주장을 소개하고 있다 [15, p. 39].

한글 문서의 어간

한글의 경우는 아무래도 이와 같을 수 없다. 《도서편집총람》 [10, p. 432]은 다음과 같이 말하고 있다.

> 우리 文은 띄어쓰기를 갖는다. 띄어쓰기의 語間은 2分을 기본으로 한다. 우리 문자는 전각문자이므로 2분의 어간은 적당하다. 그러나 이는 자간을 마이너스 치송 등으로 좁히지 않은 橫組의 경우에 그러하며, 자간을 좁힌 경우라면 어간을 2분으로 할 것인지 3분으로 할 것인지 고려하여야 한다.

이것은 그런대로 하나의 지침이 될 수는 있겠다. 그런데 두어 가지 명확히 할 것이 있다. 먼저 2분이라 함은 반각 즉 0.5em을 말함인데 양끝맞추기를 하는 상황에서는 모든 띄어쓰기 공백을 정확하게 0.5em으로 맞출 수 없다. 만약 행 끝에서 양끝맞추기를 위해 공백을 조절해야 한다면 당겨야 하는가 아니면 늘려야 하는가? 만약 당기거나 늘린다면 그 허용치는 어느 정도인가? "2분의 어간은 적당하다" 는 서술은 왜 적당한지 논증하지 않고 있다. 그것이 적당한 이유는 무엇일까?

여러 문헌을 찾아보았지만 이 이상의 지침을 발견할 수는 없었다. 워드 프로세서 아래아한글은 기본 문서 서식에서 어간을 거의 반각에 맞추고 있는 것 같은데 사용자가 별도로 옵션을 주기 전에는 이 어간을 줄이지 않는 것으로 보인다. 이것이 의미하는 것은 대부분의 어간이 실은 0.5em을 넘을 수밖에 없다는 것이다. 아래아한글의 경우 아예 상한치조차 두고 있지 않기 때문에 운이 좋으면 어간을 거의 행 전체 길이까지 늘려볼 수도 있다. 그 결과 어간이 만드는 공백으로 인해 판면은 성기고 엉성하게 보인다.

어간을 결정할 때 고려해야 할 요인은 다음과 같다.

글자의 크기 일반적으로 글자가 작을수록 어간도 작아지고 글자가 커지면 어간도 커진다.

폰트의 디자인 은바탕 글꼴의 경우 디자인 글자 폭 전체가 꽉 차 있지 않다. 이럴 때 어간을 0.5em으로 주면 실제로 자면에서 글자 영역이 차지하고 있는 부분이 글자 폭 기준 약

20. 영문 문장에서 양끝맞추기에 적용되는 또다른 개념으로 하이프네이션이 있지만 한글 문서는 하이픈을 사용하지 않기 때문에 이것은 논외로 한다.

85%라 할 때 지나치게 넓은 어간이 되어서 독서를 방해한다. 그러므로 폰트의 디자인 상태에 따라서 어간을 조금씩 변화시켜야 할 것이다.

장평과 자간 세로가 긴 장평에 마이너스 자간을 준 경우라면 정상치보다 어간을 조금 줄여야 할 것이다.

어간 허용치 양끝맞추기를 하기 위해 늘릴 수 있는 어간의 상한선을 정해두는 것이 좋다. 이 경우 좁은 행장의 문단에서 양끝맞추기에 실패할 가능성이 있지만 차라리 그것이 지나친 어간으로 공백 얼룩을 만드는 것보다 낫다.

memhangul-x는 본문 10pt를 기준으로 어간의 기본값을 '4.33pt `plus` 1.11pt `minus` 1.11pt'로 두고 있다. 이것은 대략 3.2pt에서 5.4pt까지를 허용하는 것인데 거의 반각이 평균적으로 유지되도록 하는 값이다. 은글꼴을 사용하면 이 값이 미세하게 넉넉한 느낌을 주지만 다른 글꼴을 사용하면 적당하다는 느낌이 들 것이다.

프렌치스페이싱

프렌치스페이싱(french spacing)은 이름 그대로 유럽에서의 조판 형식으로, 온점이나 물음표와 같은 마침표[21] 뒤에 추가 공백을 따로 주지 않는 방식이다. 반면에 추가 공백을 더 주는 방식은 미국에서 일반화된 조판 형식이다.

우리가 굳이 미국식을 따라야 할 필요가 있는 것은 아닐 것이다. 한중일 삼국 중에서 서양식의 피리어드(period)를 마침표 가운데 하나로 쓰는 것이 우리뿐임을 감안할 때 일본이나 중국의 방식에서는 참고할 것이 없기는 하나 일본어의 경우 고리점과 같은 모양의 종지부는 그 자체로 전각이다. 《도서편집총람》[10]에 따르면 일본에서와 마찬가지로 온점을 그 뒤의 공백까지 합쳐서 전각으로 조판한다고 하고 있다.[22]

XeTeX-k*o*는 프렌치스페이싱이 기본값이지만 `nonfrench` 옵션으로 프렌치스페이싱을 사용하지 않을 수도 있다. 문제는 이 때 추가 공백의 크기가 어느 정도여야 하느냐는 것이다. 필자는 한글 문서의 경우 미세한 `\xspaceskip`을 사용하는 것이 낫다고 생각한다. 그러나 그것은 마치 마침표와 그 뒤에 이어진 공백 전체가 전각 정도의 느낌이 나도록 하여야지 그 이상은 곤란하다. HLaTeX 1.0은 hangul 패키지의 기본값으로 무려 0.7em을 주고 있었는데 이것은 너무 과도하다.

영문 문서의 기본 `\xspaceskip`을 한글 문서에서 그대로 쓰면 마치 전각과 비슷한 효과를 얻는다. 이 값을 조정하는 것은 매우 쉬우나 행장에 따라 달라지는 추가 여분 때문에 의도하지 않은 결과가 발생할지 항상 주의하여야 한다. 필자가 제시하는 `\xspaceskip`의 값은 다음과 같다. 앞서 논의한 전각 크기의 길이를 10pt 기준으로 얻은 것이다.

```
\xspaceskip = .555em plus .07em minus .05em
```

이 값은 경험치이므로 스타일에 포함하지 않았다.

21. 한글맞춤법의 문장부호 규정에 따르면 '마침표'라는 말은 온점, 물음표, 느낌표 등 문장을 마치는 데 쓰이는 부호를 모두 묶어 부르는 말이다.

22. 영문자 피리어드를 찍고 그 뒤에 프렌치스페이싱을 두었을 때 실제로 느끼는 길이가 전각 폭일 수는 없다. 이것은 영문 글자 디자인에 좌우될 뿐 아니라 스페이싱의 폭에도 좌우될 것이다.

5.2 행간

영문 문서에서 행간격 배수(\baselinestretch)를 1.0으로 잡았을 때 10pt 문서에서 12pt 행간이 기본이다.[23] 베이스라인 기준으로 소문자 'p'의 아래끝과 소문자 'b'의 위끝이 만났다 해도 약간 떨어지는 느낌을 주기 위해서 2pt가 더 필요하다. 그러므로 영문 문서의 2배 행간(double spacing)은 행간격 배수 2.0이 아니라 1.6에 해당한다. 행간격 배수 2.0이면 행간 기준치를 12pt로 했을 때 24pt가 베이스라인 간에 주어지기 때문에 실제로 10pt 문서에서는 2배를 넘는 넓은 행간으로 짜여진다. 반면 행간격 배수 1.6은 19.2pt가 베이스라인 간에 주어지기 때문에 적절한 2배 행간이 된다. 10pt 디자인의 폰트라 해도 실제 높이는 10pt가 (당연히) 되지 않기 때문에 19.2pt 정도면 2배 행간이라 부를 수 있다. 'TeX Gyre Pagella' 글꼴의 높이(height)를 측정해보자.

The *height* of lowercase l of TeX Gyre Pagella is 7.26pt.

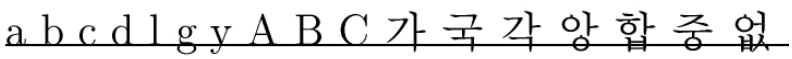
위에서 보는 바와 같이 이 길이는 약 7pt에 불과하다. 이번에는 은글꼴의 높이와 깊이(depth)를 측정해본다. 은바탕 글꼴에서 문자 박스의 높이와 깊이는 글자마다 조금씩 다를 것이기 때문에 세로 길이가 가장 긴 글자 중의 하나인 '각' 자를 이용하기로 한다.

The *height* of 각 of 은바탕 is 7.74pt, the *depth* is 1.51pt.

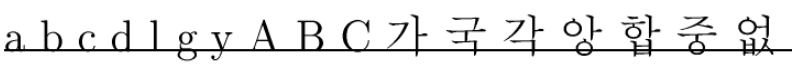
이 글자의 전체높이(totalheight)는 높이와 깊이를 합한 9.25pt가 된다. 이와 같이 한글 폰트의 자면 디자인은 대부분 영문자에 비하여 크고 자면률도 높은 편이다. 이러한 특성은 행간 값을 결정하는 데 크게 작용한다. 영문자의 경우에는 단지 소문자 몇 글자만이 베이스라인을 기준으로 아래로 내려가지만 한글은 기본적으로 1 ∼ 2pt 정도 ──영문자 베이스라인을 기준으로── 아래로 내려가 있다고 볼 수 있다. 그러므로 영문 문서와 동일한 행간을 설정하면 부득이하게 판면이 지나치게 촘촘해보이게 되고 다음 행의 시작 위치를 잘 찾을 수 없게 된다.

베이스라인

행간을 설정하기 위한 기본은 폰트의 베이스라인을 이해하는 것이다. 한글 글꼴은 베이스라인이 없다. 그러나 영문 글꼴과 비교해보면 영문 베이스라인을 기준으로 어느 정도 내려가도록 디자인되어 있느냐가 문제인데 이것이 글꼴마다 다르다. 다음은 은바탕이다.



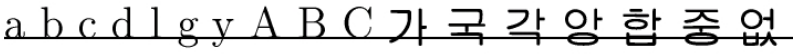
그런데 윈도 바탕 글꼴은 다음과 같다.



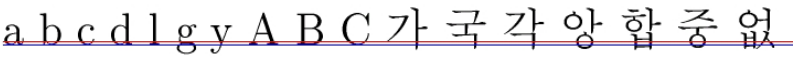
위의 예를 비교해보면 윈도 바탕 글꼴이 은바탕에 비하여 영문자 대비 조금 위로 올라가 있음을 확인할 수 있다. 문제가 되는 것은 같은 은글꼴이면서도 베이스라인을 잘못 설정한

23. memoir 매뉴얼 [14]에서는 행간을 'leading'이라고 하고 있다. 이 용어는 사진식자 조판에서 말하는 '행송'과 유사하다.

은디나루와 같은 예이다. 다음 그림에서 한글 글자들이 영문 대문자 기준으로 너무 내려와 있음을 볼 수 있을 것이다.



실제로는 영문 폰트와 한글 폰트를 섞어쓰는 경우 기준 글꼴은 한글 폰트가 되고 그 글꼴에 설정되어 있는 베이스라인 기준선을 영문자도 따라가게 된다. 예를 들어 윈도 바탕을 본문 글꼴로 쓸 때의 베이스라인과 영문 글꼴 베이스라인의 위치를 함께 표시한 다음 그림을 보라.



한글 폰트의 디자인 베이스라인이 영문 글꼴의 베이스라인보다 위에 있다. 한글이 기본 폰트인 문서에서 영문 베이스라인이 조금 아래 있는 것처럼 식자되어서 영문자 대비 한글 글자가 다른 폰트에 비해 위로 올라간 것처럼 보이게 된다.

영문자 대비 한글의 세로 위치를 어디로 잡을 것이냐는 전적으로 폰트 디자인의 문제이다. 개인적인 의견으로는 아무래도 윈도 바탕(한양 바탕) 글꼴의 위치가 우리에게 익숙하다는 것이다. 그래서 은바탕 본문으로 영문이 비교적 많이 섞이는 문서를 식자하면 한글의 수직 위치가 조금 아래로 내려가 있는 듯한 느낌이 든다.

레이아웃 프로그램은 이 베이스라인 기준선을 디자이너가 쉽게 수정할 수 있게 해준다. 그러나 텍에서는 이것을 전적으로 폰트의 특성으로 보고 있다. 만약 이 위치를 수정하고자 한다면 한글 각 문자를 식자하면서 `\raisebox`로 올리거나 내리는 방법을 쓸 수 있다.[24] 그러나 그리 해야 할 필요가 과연 있는지 의문이다.

행간의 결정요소

행간을 결정하는 데 있어서 고려하여야 할 요소는 대단히 많다. 정리하면 다음과 같다.

본문 문자의 크기 폰트 크기가 커진다면 행간도 증가하여야 한다.

폰트 패밀리 명조냐 고딕이냐에 따라, 그리고 변형서체인가에 따라 행간이 달라진다. 특히 고딕체는 명조체보다 행간을 넓게 잡아야 판면의 색조가 지나치게 어두워지지 않는다.

행장 대체로 1행의 길이가 짧으면 긴 경우보다 행간을 넓게 잡는 것이 좋다.

면적률 판형 대비 판면 면적률이 높으면 행간이 좁은 것이 좋다. 여기에는 여백 공간의 크기도 고려하여야 할 것이다.

어간 어간은 행간 결정에 대단히 중요하다. 어간보다는 행간이 넓어야 한다. 이것이 기본이고 어간이 행간보다 넓으면 필연적으로 판면을 흰색 반점이 길게 세로로 쪼개는 현상(lizard)이 발생한다.

전통적으로 행간을 결정할 때, '페이지 당 행수'라는 개념이 쓰여왔다. 이 개념의 기원은 종래 활판 시대에 판을 행별로 짜던 데서 유래하였다. 지금도 한 장의 내지 앞뒷면이 인쇄

24. 한글이나 한자를 식자할 때 각 글자박스에 hook을 줄 수 있는 hangulhook과 hanjahook 패키지를 활용할 수 있다.

되었을 때 이면에 비치는 행 흐름이 그 페이지에 인쇄된 행 흐름과 동일하면 좋다는 믿음이 널리 받아들여지고 있다. 이 방식을 한글 문서와 연관시키면 종래 전통적인 조판 방식, 즉 단 하나의 글꼴만을 사용하고 글꼴의 확대 축소는 정수배로만 하는 경우가 아니라면 실제로 구현하는 것이 꽤 어렵다.[25]

geometry 패키지는 이 페이지 당 행수 방식으로 행간을 결정하도록 하는 옵션을 제공한다. 그러나 memoir는 굳이 이런 방식을 클래스 기능으로 차용하지는 않았다. 페이지 당 행수를 기계적으로 적용하기에는 판면에서 고려해야 할 요인이 너무 많다. 그렇더라도 편집자는 인쇄된 상태에서 이면에 배어나온 페이지 색조가 독서에 어떤 영향을 미칠 것인지 고려해야 한다.

한글 문서의 기본 행간

HLaTeX 1.0과 hLaTeXp의 한글 문서 서식 스타일 hangul 패키지는 기본 행간을 1.3으로 두었다. 한글 폰트의 전체높이가 앞서 계산해본 바 있듯이 은글꼴의 경우는 대체로 8.5 ∼ 9.3pt 정도이고 한양 글꼴의 경우에는 8.7 ∼ 9.6pt 정도이기 때문에 약 9pt를 기준으로 하면 73%의 공백이 행간에 들어가는 셈이다.

워드 프로세서에서 흔히 권장하는 150% 행간이란 무엇인가를 생각해보자. 거기에서는 행간 기준치(`\baselineskip`)라는 개념이 없으므로 10pt 글꼴이라면 위행의 베이스라인에서 다음행의 베이스라인까지 15pt를 행간으로 설정하라는 의미라고 할 수 있다. 이 때는 66.7% 정도의 공백이 행간에 들어가게 된다.

만약 행 평균 전체높이의 황금비[26]에 해당하는 값을 행간으로 가지려면 전체높이가 9pt이고 폰트 크기가 10pt일 때 행간격 배수를 1.2135로 하면 된다. 영문의 경우 2배 행간의 행간격 배수가 1.6임을 앞서 본 바 있다. 한글 문서의 경우에는 이것을 1.75로 할 것을 제안한다. 이 경우 행간 거리는 21pt가 되는데 2배 행간 값으로 적절하다고 생각한다.

memhangul-x는 1.333을 기본 행간으로 하고 있다. 이 행간은 약 78%의 여백을 남기는 것으로 무게있는 글에 적당한 행간이라고 생각되어 취하였다. 도서편집총람 [10, p. 443]의 "우리 文에 있어서 문자 크기의 80%를 본문 기본 행간 크기로 함이 적당하다"고 한 권고에도 일치한다.

행간의 변화

영문 문서의 경우는 행간을 변화시킬 일이 거의 없다. 그러나 한글 문서는 자면 높이의 약 80% 정도를 추가 공백으로 넣고 있기 때문에 경우에 따라 문서의 일부에 대하여 행간을 변화시키는 일이 있다.

첫 번째로 고려해야 할 것은 각주와 플로트(float)의 행간이다. setspace 패키지는 기본 설정으로 각주와 플로트 내의 행간을 1.0으로 바꾸는 기능이 있다. 특히 2배 행간 영문 문서에서 아주 유용하다.

25. memoir 매뉴얼 [14]에서도 비슷한 취지를 말하고 있다.
26. 황금비 행간을 사용하려면 폰트 디자인 높이를 기준으로 해서는 안되고 실제 높이를 기준으로 해야 한다.

문제는 한글의 경우 1.0 행간이 항상 적절한 것은 아니라는 것이다. 당연히 각주와 플로트 내의 행간이 본문 행간과 동일하다면 각주 글꼴이 본문보다 조금 크기가 작은 점을 감안할 때 지나치게 백면이 넓어져서 페이지 색조가 흐려지는 문제점이 발생한다. 또한 각주의 성격상 본문보다 촘촘하게 조판하여도 문서의 전체적인 균형이 흐트러지지는 않는다.

memhangul-x는 각주와 플로트의 행간격을 별도로 지정하여 변화시킬 수 있게 하고 있다. 기본값은 본문 행간이 1.333인 데 비해 각주와 플로트 행간을 1.1로 지정한 것이다. 이따금 이 행간이 너무 좁지 않은가 생각할 때도 있으나 그것은 사용자가 선택할 문제이다.

두 번째로 고려해야 할 것은 `verbatim` 환경의 행간이다. memoir는 `verbatim` 환경의 글자 크기를 본문보다 조금 작게 설정해두었으므로 역시 `verbatim` 환경의 행간을 본문과 동일한 행간으로 조판하면 문제가 생긴다. 그리고 `verbatim` 환경의 목적이 주로 소스를 소개하거나 하는 것이고 대부분이 영문자로 이루어진 텍스트일 가능성이 높기 때문에 한글 문서 기본 행간을 적용하기에는 적절하지 아니하다. memhangul-x는 이 역시 각주와 플로트 행간과 동일하게 '좁은 행간'이 적용되는 것을 기본값으로 하여 두었다.

마지막으로 ──논란거리라고 생각하지만── `quote` 또는 `quotation` 환경에서 좁은 행간을 적용하는 문제가 있다. 《도서편집총람》[10, p. 445]에서,

> 구획행간으로 구획될 구획문은 본문 크기의 문자이거나 보다 작은 크기의 문자로써 본문 행간과 같은 크기의 행간으로 하거나 아니면 본문의 기본행간과는 달리 보다 좁은 행간으로 하게 된다. 구획문의 행간을 좁히는 방법의 타당성은 그 구획문이 행수가 많지 않고 작은 문자일 때 구획문의 좁은 행간이 오히려 판면에 안정을 주고 구획을 시각화하며 판면에 질감을 준다는 데 있다.

라 하여 행간을 줄이는 인용문이 타이포그라피상 허용되는 방법임을 말하고 있다.

일부 문단에 행간을 변경하는 방법은 한 가지 문제를 불러일으킨다. 그것은 원칙적으로 한글 문서의 모든 행간이 동일한 간격을 유지해야 이면(裏面) 비침의 어긋남이 발생하지 않는다는 요구와 충돌한다는 것이다.

6 자간 · 장평 · 어간 · 행간 조절

이제 앞 절에서 논의한 타이포그라피 요소들을 지정하거나 변경하는 방법을 알아보자.

6.1 자간의 설정과 조정

간격에 관한 문제는 거의 대부분 XeTeX-k*o*의 기능으로 구현하며, 폰트의 속성으로 취급한다. 예를 들어 다음은 본문 글꼴의 자간을 −0.05em으로 하라는 의미이다.

```
\setmainhangulfont[interhchar=-.05em]{나눔명조}
```

6.2 장평의 설정과 조정

fontspec 패키지의 옵션을 이용하여 본문 글꼴의 장평을 90%로 주려면 다음과 같이 한다.

```
\setmainhangulfont[FakeStretch=.9]{나눔명조}
```

임의로 장평을 '자유롭게 조절' 한다면 오히려 그것이 문제가 될 수 있다. 폰트 디자인은 디자이너의 의도를 반영하여 사용되어야 할 것인데 임의로 장평을 줄이는 것은 그것을 사용자가 무단 변경하는 것에 지나지 않는다. 따라서 장평의 사용은 자신의 디자인에 확신이 있는 경우가 아니면 사용하지 말아야 한다.

6.3 어간의 설정과 조정

어간은 \spaceskip과 \xspaceskip 값에 의하여 결정된다. memhangul-x는, 즉 XeTeX-k*o* 는 프렌치스페이싱이 기본값이다. 그러나 프렌치스페이싱을 적용하고 싶지 않을 경우에는 nonfrench 패키지 옵션을 주면 된다.

6.4 행간의 설정과 조정

텍에서 행간을 결정하는 길이 값은 \baselineskip이라는 간격 변수이다. 그리고 이 값은 폰트 선택 명령인 \fontsize 명령의 두 번째 인자로 주어지는 값이기도 하다.

```
\fontsize{10}{20}\selectfont
```

위의 명령은 폰트 크기를 10pt로, 그리고 이 폰트의 행간을 20pt로 설정한다. 이와 같이 행간은 폰트 속성으로 처리하도록 되어 있기 때문에 사용자가 직접 \baselineskip을 바꾸면 뜻하지 않은 결과를 초래할 수 있다. 이 때문에 라텍에서는 기본적으로 폰트 속성의 일부로서 행간을 먼저 정해준 다음 이것의 배수를 이용하여 행간을 바꾸도록 하고 있는데 그 배수 값이 \baselinestretch이다. 이 값은 포인트 단위의 길이값이 아니고 \baselineskip에 곱하여질 상수이므로 \renewcommand로 그 값을 변경한다. \linespread는 주로 프리앰블(preamble)에 쓰여서 문서 전체의 \baselinestretch 값을 지정한다. 문서 중간이나 프리앰블 모두에 쓰일 수 있는 명령으로 setspace 패키지가 제공하는 ──그러므로 memhangul-x에서도 쓸 수 있는[27] ── \setstretch가 있다.

행간 설정 명령과 선언

문서 기본 행간을 변경하려면 \linespread나 \setstretch를 사용하는 것이 라텍의 일반적인 방법이다.

```
\linespread{1.333}
```

memhangul-x에서 기본 행간을 변경하는 또다른 방법이 있다. 그것은 \SetHangulspace라는 명령을 사용하는 것이다. 이 명령은 패키지 옵션 nosetspace이 설정된 경우에는 큰 효력이 없다. 사용법은 다음과 같이 두 개의 상수 인자를 지정하는 것이다. 두 상수는 모두 \baselinestretch 값을 의미한다.

```
\SetHangulspace{1.333}{1.2135}
```

27. memhangul-x는 setspace 패키지를 그대로 포함하고 있다. 따라서 setspace를 별도로 불러들이면 안된다.

첫 번째 인자는 본문 행간에, 두 번째 인자는 '좁은 행간'에 적용되는 행간 배수이다. '좁은 행간'은 각주와 플로트, 그리고 `adjustquotespacing` 상태라면 `quote`와 `quotation` 환경에 적용된다. 이 명령은 프리앰블에서 사용한다. 즉 문서 전체에 대한 명령형 선언이다.

이 값을 문서 중간에 바꾸려면 다음 두 매크로를 사용한다.

```
\ResetHangulspace{1.2}{1.1}
\RestoreHangulspace
```

첫 번째 `\ResetHangulspace`의 사용법은 `\SetHangulspace`와 같다. 다만 문서 중간에 사용한다는 점만 다르다. `\RestoreHangulspace`는 프리앰블에 선언된 ──또는 기본값의── 행간 설정으로 되돌리는 선언이다.

`verbatim`, `epigraph`, `verse` 환경의 행간

`verbatim` 환경의 행간은 제어 문자열 `\hangulverbatimspacing`이 통제한다. 이 값을 바꾸려면 제어 문자열을 직접 재정의하지 말고 `\SetHangulVerbatimSpace` 명령을 이용한다.

```
\SetHangulVerbatimSpace{1.0}
```

`epigraph` 환경의 행간은 제어 문자열 `\epigraphspacinghook`에 의해 통제된다. 이 값을 바꾸려면 제어 문자열을 직접 재정의하지 말고 `\epigraphspacing` 명령을 이용한다.

```
\epigraphspacing{1.0}
```

이 결과는 `\epigraph` 명령 및 `epigraphs` 환경에 모두 적용된다.

`verse` 환경과 `altverse` 환경, `symbols` 환경의 행간은 `quote` 환경의 행간 설정을 따른다. 만약 다른 행간을 가지게 하려면 `\adjustquotespacing`과 `\noadjustquotespacing` 선언을 적절히 활용하라.

임의의 행간

memhangul-x에서는 setspace의 모든 기능을 그대로 쓸 수 있다. 가장 범용성이 높아서 일반적으로 행간을 조절하는데 쓰는 것은 `Spacing` 환경이다. 이 환경은 한 개의 인자를 가지는데 그 값으로 환경 유효 범위 내의 행간을 조절해준다.

```
\begin{Spacing}{1.5}
running text...
\end{Spacing}
```

임의의 행간을 별도로 설정하고자 할 때는 이 환경을 사용하면 될 것이다.

6.5 `finemath`와 폰트 속성으로서의 미세 간격

ko.TEX의 기능 중 `finemath`라 불리는 것은 문장 중의 수식과 그 뒤에 이어지는 글자 사이에 약간의 공백을 추가하여 수식의 가독성을 높이려는 기능이다 [4].[28] XETEX-k*o*와 oblivoir는

28. `finemath`의 구현과 더불어 시작된 한글 타이포그래피에 있어서의 간격 문제는 이어지는 XETEX-k*o*와 LuaTEX-k*o*에 와서 완성되었다.

표 7. 폰트 속성으로 미세 간격 조절

hu	영문자와 한글 사이, 영문괄호와 한글 사이(2배)
interhchar	한글 자간
lowerperiod	한글 문자에 이어지는 마침표를 끌어내림
lowerquestion	물음표를 끌어내림
lowerexclamation	느낌표를 끌어내림
lowercomma	쉽표를 끌어내림
preperiodkern	마침표 앞에 두는 간격
postperiodkern	마침표 뒤에 두는 간격
prequestionkern	물음표 앞에 두는 간격
postquestionkern	물음표 뒤에 두는 간격
preexclamationkern	느낌표 앞에 두는 간격
postexclamationkern	느낌표 뒤에 두는 간격
precommakern	쉼표 앞의 간격
postcommakern	쉼표 뒤의 간격
interpunctskern	마침표, 쉼표, 물음표, 느낌표 사이의 간격 지정
quotewidth	영문 인용부호의 폭
postmathskip	수식 직후의 간격

따로 finemath 관련 옵션을 두고 있지 않으나, 동일한 기능이 폰트 속성에 간격을 부여하는 방식으로 변화하였다.

```
\setmainhangulfont[postmathskip=0.2em]{나눔명조}
```

이외에 미세간격을 조정하는 옵션이 마련되어 있으며 이를 요약한 것이 표 7이다. 이 가운데 마침표, 쉼표, 물음표 등 문장부호를 끌어내리거나 앞뒤로 간격을 두는 문제는 [2]에서 연구된 문제를 발전시켜 구현한 것이다.

7 면주와 페이지 스타일

페이지 스타일이란 면주[29]양식을 말한다. 먼저 면주 양식을 설계할 때 고려해야 할 것을 생각해보자.

7.1 면주에 관련된 몇 가지 고려사항

면주를 붙일 수 있는 곳은 판면의 상단과 하단이다. 이따금 판면의 좌우를 면주 영역으로 사용하는 경우도 있으나 이것은 디자이저가 특별한 의도를 구현하기 위하여 사용하는 것이고 일반적으로 권장할 만한 것은 아니다.

29. 면주를 '하시라' 라고 하는데 이것은 일본어 '柱' 의 훈독이다. '면주' 라는 말이 이 일본어에서 온 것으로 알고 있다. 차라리 '난외표제' 라 함은 어떨까?

상단 면주와 하단 면주

상단 면주를 사용할 것인가 하단 면주를 사용할 것인가? 면주의 기능이 독자로 하여금 현재의 위치를 파악하게 하고 검색을 돕는 데 있다면 상단이든 하단이든 일관성만 있으면 될 것이다. 라텍의 book 클래스의 기본값은 `headings` 페이지 스타일인데 이것은 장이 시작하는 페이지는 면번호만을 하단 중앙에 넣고 다른 페이지는 상단 면주를 사용하는 방식으로 되어 있다. 책에서 가장 익숙한 방식이다. 장이 시작하는 쪽에도 상단 면주를 넣게 되면 장 표제와 면주가 충돌하는 느낌이 들기 때문에 바람직하지 않다. 장 표제면에는 페이지 번호만을 넣는 것이 통상이고 그것으로 충분하다. 하단 면주를 사용하는 경우는 면주의 기능이 상대적으로 약화되는 경향이 있고 페이지가 아래로 쳐져 보일 수 있으므로 판면을 조금 위로 올린다든지 하여 보정할 필요가 있다.

상하단에 모두 면주를 넣는 경우도 있다. 이 때 상단에는 보통 일반적인 표제를 붙이고 하단에는 면번호를 넣기도 하고 상단에 표제와 면번호를 모두 넣고 하단 면주에는 저작권 사항이나 기타 정보를 넣기도 한다. 상하단을 모두 면주로 사용하는 것은 자칫하면 판면이 불안정해지고 지나친 장식의 느낌이 날 수 있으므로 최소한으로 사용하는 것이 좋을 것이다.

면주까지의 거리

편집 영역에서 면주까지는 어느 정도의 거리를 두어야 할까?

상단 면주는 `\uppermargin` 영역 안에 들어가야 한다. 면주에서 편집 영역까지의 거리는 `\headsep`으로 지정된다. 이 거리는 지나치게 멀면 본문으로부터 동떨어져 보여서 좋지 않고 너무 붙으면 본문과 구별이 잘 되지 않는다. 보통 권장하는 길이는 본문의 1행간을 비우는 것인데 기본 문서에서 12pt가 행간 기준치이므로 1행간 16pt에 행간 기준치를 더하면 조금 떨어지는 느낌이 날 수 있다. 대신 2배각 세로 길이(2em) 정도이면 적당하지 않은가 한다. 본문을 10pt 기본값으로 작성하고 행간을 1.333으로 정한 경우 memoir 기본값인 약 18pt는 1.5배 행간에 해당하고 2em보다는 조금 작은 정도이다. 이 기본값을 그대로 사용하는 것도 나쁘지 않다고 본다.

문서를 `\raggedbottom`으로 조판하는 경우 하단 면주를 사용하는 것은 다음 두 가지가 고려되어야 한다. 첫째는 각주의 양과 유무이다. 각주가 있을 경우 각주로부터 면주까지의 거리에 신경을 써야 한다. 두 번째는 판면의 아래끝이 균일하지 않은 상태에서 페이지마다 면주의 위치가 다르게 보이면 일체감을 잃을 수 있다는 것이다. 하단 면주를 일관되게 사용하려면 각주가 없거나 거의 없는 문서 또는 판면 아래끝이 비교적 일관되게 놓이는 문서가 적당하다.

면주의 범위와 괘선

면주의 폭은 판면폭(행장)과 동일하게 설정하는 것이 일반적이고 이것이 깔끔하고 단정해 보인다. 그런데 'The LaTeX Companion' [12]와 같은 책은 상단 면주의 폭이 바깥쪽 여백 범위까지 침범하도록 디자인하고 있다. 이 책에서는 여백 영역까지 벗어나는 표와 그림이 상당히 많은데 만약 면주 영역을 판면폭으로만 제한하면 페이지 상단에 위치한 플로트의

폭보다 면주폭이 좁아져서 페이지의 균형감을 잃게 된다. 그러므로 넓은 표나 그림이 많이 사용되는 문서에서는 이러한 여백까지 침범하는 면주 영역을 생각해보는 것이 좋다. `Ruled` 페이지 스타일은 이와 같은 넓은 면주 영역을 설정하고 있다.

면주에 괘선을 그리는 것도 책 [12]의 특징이다. 페이지 스타일 `companion`과 `ruled` 및 `Ruled`는 모두 괘선을 면주 밑에 긋고 있다. 생각건대 한글 문서의 경우 괘선을 남용하는 것은 촌스럽게 보일 위험을 다분히 내포한다. 필요하지 않은 괘선은 긋지 않는 것이 좋다.

면주의 내용

면주에 페이지 번호 즉 면번호[30]는 반드시 들어가야 한다. 그리고 면번호 이외에 '면색인' — 면주 표제(running heading) — 이 포함되는 경우가 많다. 작은 규모의 문서에는 면번호만 있고 면색인이 없는 경우도 있다.

면주 표제에 무슨 내용을 넣어야 할 것인가?

1. 책 이름이나 문서 전체의 제목은 넣지 않는 것이 좋다. 왜냐하면 면주가 '색인'의 기능을 하고 있다 할 때 처음부터 끝까지 동일한 내용이 적혀 있다면 이것은 색인으로서의 기능이 없는 것이나 다름없기 때문이다.
2. 왼쪽면에는 조금 더 고정적인 요소를 오른쪽면에는 조금 더 세분된 요소를 표시하는 것이 일반적이다. 즉 편-장 체제라면 왼쪽면에는 편표제 오른쪽면에 장표제를 싣고 장-절 체제라면 왼쪽면에 장표제 오른쪽면에 절표제를 싣는다.
3. 면번호를 판면의 바깥쪽에 찍는다. 그렇게 하는 이유는 색인으로서의 면번호가 페이지를 완전히 펴지 않은 상태에서도 찾아볼 수 있게 하기 위함이다. 안쪽에 페이지 번호를 두는 것은 좋지 않다.
4. 면번호와 면색인은 서로 독립적으로 취급하여도 좋다. 우리나라 출판물에서 많이 취하고 있는 방법은 왼쪽면에는 왼쪽끝에 면번호를 두고 약간의 간격과 면색인을 적은 다음 왼쪽면의 오른쪽끝은 비우는 것이고 오른쪽면은 왼쪽끝을 비우고 오른쪽 끝에 면색인과 면번호를 차례로 두는 방법이다. 필요하다면 면색인을 가운데 두어도 좋겠지만 이것은 좌우 페이지 구분이 불필요할 때 사용할 수 있다.
5. 면색인의 길이는 행장의 반을 넘지 않는 것이 좋다. 만약 표제행이 길다면 적당한 위치에서 줄인다.

위의 4번은 영문 책의 경우와 관행에 차이가 있다. 예컨대 라텍 표준 book 클래스의 기본값인 `headings` 페이지 스타일은 면번호를 바깥쪽에 두는 것은 동일하지만 면색인은 그 페이지의 면번호 반대편에 둔다.

면주의 서체와 크기

면주는 본문과는 별개인 고정적 디자인 요소이므로 본문 서체를 따를 필요는 없다. 영문 문서의 경우 작은 대문자나 대문자의 이탤릭 (또는 기울인) 서체를 면색인에 사용하고 면번호는

30. folio. 어떤 책에서는 '면숫자'라 부르기도 한다.

평범한 서체로 식자한다.

우리나라 책은 상단 면주의 경우 폰트 크기를 줄이는 것이 관행인 듯하다. 예컨대 책 [10]의 경우는 본문보다 2급 정도 작은 폰트로 면색인을 작성하는 대신 면번호는 산세리프의 보통 크기로 찍고 있다.

memhangul-x가 memoir의 페이지 스타일을 일부 한글화하였다고는 하나 면주 서체를 크게 변경하지는 않았다. 그 결과 다음과 같은 문제가 생겨난다.

- `headings` 스타일에서 `\textsl`로 지정된 부분은 기울인 글꼴로 식자될 수 있다. 만약 `gremph` 옵션을 준 경우라면 이번에는 그래픽 글꼴로 식자된다. `\emph` 명령으로 구현되는 서체가 면색인에서 사용되는 것이 바람직한가는 전혀 확신할 수 없다. 그러므로 이것은 기정값을 그대로 쓰지 않도록 수정되는 것이 바람직할 것이다.
- 서체의 크기가 본문의 크기와 동일하다. 이 때문에 면주가 두드러져 보이는 효과가 발생한다.
- `companion` 스타일을 쓰면 면주가 굵은 글꼴로 식자된다. 특별한 경우가 아니면 면주에 굵은 글꼴을 쓸 필요는 없을 것이다.

이러한 문제를 고려하여 memhangul-x는 `hangul`이라는 한글화 페이지 스타일을 추가로 제공한다. 이것은 면색인의 서체 크기를 `\small`로 하고 면번호를 `\normalsize`의 산세리프로 찍도록 한 것이다.

페이지 스타일을 정의하는 것은 어려운 일이 아니므로 디자이너가 직접 자신의 페이지 스타일을 정의해서 쓰는 것이 좋다.

면주와 면번호 붙이기

면주를 붙이지 않는 본문 페이지를 '숨김 페이지(drop folio)'라 한다. 내제지, 헌사, 삽화 도판면, 별지삽입면, 접어넣은 페이지, 판권지는 숨김 페이지가 된다. 또한 새로운 장을 시작하기 전의 공백면에도 면주를 붙이지 않는다. memoir 클래스는 이러한 숨김 페이지를 대부분의 경우 잘 처리해준다. 그러나 필요하다면 사용자가 직접 `\pagestyle{empty}`를 지시해주어야 하는 경우도 있다.

면번호 붙임의 체제는 frontmatter에 로마자 면번호를 붙이고 mainmatter 본문부터 새로이 1 페이지로 시작하여 책의 마지막까지 ──즉 backmatter에 별도의 페이지 번호를 매기지 않는다── 아라비아 숫자로 일련번호를 붙인다. 중간에 접어넣은 페이지나 별지삽입면, 삽화도판면은 면번호를 붙이지 않을뿐 아니라 페이지 매김도 하지 않는 경우가 있다.[31] 우리나라 책은 frontmatter에도 아라비아 숫자로 면번호를 붙이기도 한다. 그래도 본문이 시작하는 mainmatter부터 새로이 페이지 번호를 시작하는 것이 관행이다. 만약 frontmatter의 면번호를 로마 숫자로 하지 않고 아라비아 숫자로 표시하고 싶다면 `arabicfront` 옵션을 memhangul-x에 지시한다.

31. 그러나 온라인 문서의 경우라면 이런 페이지에 대해서도 페이지 매김을 해주어야 한다.

표 8. memoir의 페이지 스타일

페이지 스타일	형식
`empty`	머릿글과 바닥글에 아무것도 표시하지 않는다.
`plain`	머릿글은 비우고 페이지 번호는 페이지 바닥에 중앙 정렬된다.
`headings`	바닥글을 비우고 머릿글에는 페이지 바깥쪽에 페이지 번호와 장 또는 절의 표제를 붙인다.
`myheadings`	`headings` 스타일과 같지만 머릿글에 올 표제를 지정할 수 있다.
`ruled`	바닥글의 바깥쪽에 페이지 번호를 붙인다. 머릿글에는 장 또는 절의 표제를 넣고 머릿글 아래에 선을 긋는다.
`Ruled`	`ruled` 스타일과 비슷하지만 머릿글과 바닥글 범위가 여백까지 연장된다.
`companion`	책 [12]의 페이지 스타일을 흉내낸다.
`part`	`plain` 스타일과 같으나 `\part` 명령이 불릴 때 적용된다.
`chapter`	`plain` 스타일과 같으나 `\chapter` 명령이 불린 페이지에 적용된다.
`cleared`	`empty` 스타일과 같으나 `\clear...` 명령이 불릴 때 필요한 백면에 적용된다.
`title`	`plain` 스타일과 같으나 `\maketitle` 명령이 불릴 때 적용된다.
`titlingpage`	`empty` 스타일과 같으나 `titlingpage` 환경이 적용되는 페이지에 적용된다.

7.2 페이지 스타일

면주 설정을 위한 fancyhdr라는 탁월한 패키지가 있다. 라텍 표준 클래스를 쓴다면 이 패키지를 이용하여 원하는 거의 모든 설정을 할 수 있다. memoir는 이 면주 설정 기능을 자체 지원하고 있다.

memoir 클래스에서 제공하는 페이지 스타일은 표 8과 같다. 위에서 네 개의 페이지 스타일은 라텍 기본 페이지 스타일과 동일하다.

memhangul-x는 `headings`, `ruled`, `Ruled`, `companion` 페이지 스타일을 한글 환경에 맞게 수정하여두고 있다. `empty`와 `plain` 스타일은 수정없이 사용할 수 있는 것이므로 결국 `myheadings`만이 재정의되지 않은 셈이다. memhangul-x에서 아무런 페이지 스타일 설정 없이 쓰면 `myheadings`를 사용한다. 그 결과 한글 문서 면주에 '제 1 장'과 같이 나타나지 않고 '장 1.'과 같이 나타날 수 있다. 이것을 수정하려면 페이지 스타일을 `headings` 또는 `hangul`로 선언한다.

```
\pagestyle{hangul}
```

한글 문서에서 `headings` 또는 `myheadings`를 사용할 때 발생할 수 있는 문제에 대해서는 앞서 이미 논의하였다. 문서 기본 페이지 스타일로 `hangul`을 사용하려면 `hangulpagestyle` 옵션으로 memhangul-x를 부른다.

페이지 스타일을 정의하기 쉽다는 것이 memoir의 장점이다. 새로운 페이지 스타일을 (재)정의하려 할 때 필요한 매크로 목록을 표 9에 요약하였다.

새로운 페이지 스타일을 설정하는 구체적인 예제가 memoir 매뉴얼 [15, p. 177f]에 있다. 이 예제를 한번 숙독하는 것으로도 충분히 원하는 모양을 만들 수 있을 것이다.

표 9. 페이지 스타일 정의를 위한 명령

명령	인자	설명
\makepagestyle	style	새로운 스타일을 설정한다.
\aliaspagestyle	alias, orig	이전 스타일을 개명한다.
\copypagestyle	new, orig	스타일을 복사한다.
\makeevenhead	style, l, c, r	짝수쪽 상단면주를 설정한다.
\makeoddhead	style, l, c, r	홀수쪽 상단면주를 설정한다.
\makeevenfoot	style, l, c, r	짝수쪽 하단면주를 설정한다.
\makeoddfoot	style, l, c, r	홀수쪽 하단면주를 설정한다.
\makerunningwidth	style, len	머리글의 너비가 len이 되도록 설정한다.
\makeheadrule	style, wd, thick	머릿글 아래에 선을 긋는다.
\makefootrule	style, wd, thk, sk	바닥글 위에 선을 긋는다.
\normalrulethickness		선의 두께를 기본값으로 한다. (0.4pt)
\footruleheight		이 값이 0이면 선이 보이지 않는다.
\footruleskip		편집영역 하단과 바닥글 사이의 선이 그어질 위치 확보
\makeheadposition	style, eh, oh, ef, of	각각의 면주 정렬 위치를 설정한다.
\makepsmarks	style, code	면주에 매크로를 추가하거나 동작시킨다.

7.3 페이지 아래끝 맞추기

원래 텍은 수학식을 조판하기 위해 만들어졌다고 해도 과언이 아니며 지금까지 수식을 미려하게 조판하는 능력이 높이 평가되어 왔다. 텍을 수식 조판기 정도로만 생각하는 것은 적절하지 않겠지만 수식 조판을 위주로 설계된 것이 남긴 결과 중에는 '문단 간 가변폭 간격' 이라는 독특한 특징이 있다. 별행 수식이 들어오는 경우 페이지 아래끝 맞추기를 위해서 문단 사이의 폭을 적당히 늘이거나 줄이는 것이다. 이것을 \flushbottom이라 하는데 텍의 기본값으로 설정되어 있다.

생각건대 한글 문서에 있어서 전통적인 조판 방식은 설령 페이지 아래끝을 맞추지 못하더라도 문단 간 간격은 고정시키는 것인 듯하다. 영문 문서와는 달리 한글 문서의 행간은 상당히 넓어서 만약 문단 간 간격을 일관되게 유지하지 아니하면 그 차이가 금방 식별된다. 그리고 영문 문서의 경우 1행간을 넣기 위해 필요한 간격이 10pt 본문의 경우 12pt에 불과하지만 한글 문서의 경우는 16pt가 필요한 것이다. 따라서 문단 간 간격은 의도보다 훨씬 벌어져버리고, 이것은 경우에 따라 독서를 방해하고 글의 흐름의 일관성을 끊어놓을 우려가 있다.

텍 명령 \raggedbottom은 페이지 아래끝 맞추기를 해제한다. 아래끝이 가지런하지 않더라도 페이지 나눔을 행하도록 강제하는 것이다. \raggedbottom 선언은 페이지 아래끝을 가지런하게 맞추지는 못하지만 문단 간 간격은 일정하게 유지시켜준다. 이 두 가지를 동시에 충족하는 것은 매우 어렵다. 문제는 의도하지 않은 \raggedbottom의 효과이다. 즉 다음 한 행 정도가 충분히 페이지 아래에 올 수 있을 것 같은데 다음 페이지로 넘어가고 페이지 아래가 비는 경우를 흔히 만나게 된다. 이와 관련된 몇 가지 문제를 검토해보고자 한다.

1행 1음절 금칙 문제

만약 '고아'를 1행 1음절 문제로 이해한다면 문장이 끝나는 "-다." 만이 새로운 줄로 식자되는 것을 금지해야 한다는 일부 주장을 검토해야 한다. 논의를 명확히 하기 위해 조금 부연하면 이것은 "-다." 앞에서는 개행하면 안된다는 것을 의미하는 것이 아니고 "-다." 앞에서 개행이 일어나서 그것으로 문단이 끝나는 경우이거나 이것이 이른바 '과부'가 되는 경우를 금지해야 한다는 정도로 받아들일 수 있다.[32] 실제로 국내의 유수한 출판사에서 출판된 책들을 보면 문단 중간의 문장에서 "-다." 개행은 허다하게 발견할 수 있다. 아마도 이것을 금칙 처리하자는 주장은 단 한 글자만으로 한 행을 차지하는 것이 미관상 보기가 그다지 좋지도 않을 뿐더러 한 단락이 끝나는 마지막 문장의 마지막 글자 앞에서 개행하면 독서의 흐름을 방해할 수 있다는 생각에서 비롯된 것이 아닌가 한다.

그러나 필자의 생각은 다르다. 새로운 페이지에서 외따로 과부로 남는 것은 반드시 피해야 하지만 한 페이지 안의 문단에서 "-다."로 끝나는 줄이 과연 큰 문제거리가 되어야 하는가 의심스럽다. 한글은 하이픈이 없는 대신 글자 단위로 어디에서나 끊을 수 있다는 점과 모순된다. 이것은 영문 문서에 비기자면 마지막 행 하이픈 기피와 일맥상통한다. 서양말의 경우는 하이픈이 오히려 단어의 지각을 해칠 수 있지만 문장 종결을 위한 "-다."는 오해의 소지도 없고 이것이 다음 줄로 넘어갔다고 해서 독서의 방해를 받을 것 같지는 않다. 이것을 기피하자는 주장은 아마도 미학적 이유에서 비롯된 것이 아닌가 한다. 그리고 이 문제는 우리가 논의하고 있는 고아-과부 문제와는 별 관계가 없다. memhangul-x는 이 문제에 대해서 아무런 조치도 취하고 있지 않으므로 아마도 경우에 따라 1행 1음절이 발생할 수 있을 것이다.

그럼에도 불구하고 1행 1음절이 그다지 바람직하지 않은 판면을 만드는 것은 사실이다. 무엇보다도 화이트 스페이스가 너무 커져서 판면의 색조가 엷어지는 것을 피할 수 없다. 그러므로 이를 '금칙'으로까지 간주할 필요는 없겠으나 타이포그라퍼의 입장에서는 회피하고 싶은 것임에 틀림없다고 생각한다. 앞으로 더 많은 연구가 필요하다고 하겠다.

장절 표제 외톨이줄 금지

이따금 절의 표제만이 앞 페이지에 남고 그 절의 문단은 모두 다음 쪽으로 넘어가는 경우가 있는데 이것은 아주 전형적인 고아 현상으로 보아야 할 것이다. 왜냐하면 장절의 표제는 당연히 그 뒤에 이어지는 장절 문장을 기대하고 있는 것으로 하나의 독립적 문단을 이룰 수 없기 때문이다. 일부 워드 프로세서가 처리하는 방식은 '페이지의 아래끝을 맞추는 것'을 '장절 표제 고아를 금지하는 것'보다 우선시해서 장절 표제든 뭣이든 좌우간 일단 페이지 아래끝까지 모두 식자하고서야 다음 페이지로 넘어가는데 이는 대단히 무책임한 방식이라고 생각한다. 장절 표제는 결코 앞 페이지에 외따로 남아서는 안된다. 설령 그 앞 페이지의 하단이 한 행 이상의 여유분이 남아서 채워지지 않는다 하더라도 마찬가지다.

memoir는 이러한 경우 앞 페이지에 남는 여분 공백을 어떻게 처리할 것인지 지정할 수 있다. `\raggedbottomsectiontrue`는 다음 페이지로 장절 표제를 옮긴 후에 남는 여분의 공백

32. '고아'나 '과부'와 관련된 개념상의 혼란으로 이 문제를 이 글에서는 더이상 다루지 않는다. 여기서 '과부가 되는 것을 금지'한다는 말은 새로운 페이지의 첫 줄이 1음절 1행으로 문단이 종료되는 것을 금지해야 한다는 의미이다.

표 10. 항목 체계의 깊이

−1	0	1	2	3	4	5
\part	\chapter	\section	\subsection	\subsubsection	\paragraph	\subparagraph

을 \raggedbottom으로 식자하도록 설정하는 것이다. \raggedbottomsectionfalse는 이것을 \flushbottom으로 식자하도록 설정하는 것인데 이 때 길이값 \bottomsectionskip에 의하여 페이지 바닥에 허용되는 스트레치의 길이를 정해줄 수 있다. 이 값이 0이면 페이지 바닥까지 앞 페이지의 글줄이 \flushbottom 처리될 것이다.

8 장절 표제

하나의 책은 여러 부분으로 나누인다. 이 나누는 체계를 편장절 체계 또는 항목 체계라 한다. 이 편장절에 적절한 번호와 항명칭을 붙이는데 이것을 항번이라 한다. 이 절에서는 항목 체계를 구성하고 항번을 붙이는 방법, 그리고 장절 표제[33]를 디자인하는 방법에 대해서 논의한다.

8.1 항목 체계

논문이나 논저의 경우는 말할 것도 없지만 크지 않은 규모의 보고서들도 일정한 편장절 체계를 갖추는 것이 보통이다. 편장절 체계는 자신이 전달하고자 하는 내용을 가장 효과적으로 전달할 수 있도록 짜여지면 된다.

항목 체계의 깊이

일반적인 도서에서 편-장-절 체계를 채택하는 경우 항목 체계의 깊이는 글의 성격에 따라 다르지만 대체로 5단계 내외로 구성하는 것이 통상이다. 라텍의 book 클래스와 memoir 클래스에서는 표 10에서 보인 바와 같은 7단계 구성을 하고 있다.[34]

원칙적으로 항목 체계는 지나치게 복잡하면 안된다. 독자가 현재 읽고 있는 위치를 잘못 파악하게 하는 것은 좋은 문서 구성이 아니다. 그러므로 가능한 최소한으로 항목 체계의 깊이를 정하는 것이 필요하다.

일반적인 논저의 경우라면 \subsubsection 이후를 항목으로 처리하는 것은 바람직하지 못하다고 생각한다. 라텍의 장절명령 가운데 \paragraph는 매우 독특한 이름을 가지고 있지만 어엿한 항목 체계의 일부를 나타내는 명령이다. 그러나 \paragraph를 항목 체계로 간주하지 않는 것이 좋을 것이다. \subsubsubsection이 왜 없는지 궁금해하는 사람들이 있다. 그러나 \subsub...과 \sub... 만으로도 헷갈리는 데는 충분하다. 여기에 새로운 요소를 추가하여 문제를 복잡하게 만들 것이 무엇인가? 아무튼지 만약 자신이 쓰고 있는

33. 항번은 '번호'를 말하는 것이고 장절 표제 문단은 항번이 붙고 장절의 구획을 표시하면서 구획의 내용을 전달하는 제목 텍스트를 말하는 것이다. 경우에 따라 장절 표제 없이 항번만을 남기거나 심지어 항번도 남기지 않고 공행으로 장절을 구획하는 경우도 없지는 않지만 이 글에서는 논외로 하겠다. 즉 원칙적으로 모든 장절에는 제목(표제)이 붙는다고 가정하자. 항번과 장절 표제 문단, 이 둘을 합쳐서 '장절 표제'라고 부른다.

34. 라텍의 article 클래스는 이와 조금 다르지만 이 글에서 다루지는 않는다.

글에서 \subsubsubsection 또는 \subsubparagraph가 필요하다고 느끼는 순간이 온다면 혹시라도 항목으로 다루어져서는 안될 것을 항목으로 취급하고 있지는 않은지, 자신의 글의 구성이 과연 적절한지 스스로 돌이켜볼 필요가 있다.[35]

단순한 사실의 나열이라면 장절 항목으로 취급하려 하여서는 아니될 것이다. 다음 소절에서 장절 항목과 장절 항목이 아닌 것을 구별해보기로 하겠다.

항목 체계로 다루어질 수 없는 문단들

다음과 같은 문단은 항목 체계로 다루어질 수 없다. 즉 장절명령을 이용하여 표현하여서는 안된다. 이것은 특히 경험이 없는 저자들이 자주 저지르는 실수로서 좋지 않은 판면과 문서 구성이라는 결과를 가져오곤 한다.

열거나 예시 단순한 사실의 나열이나 이미 기술된 내용을 한 항목씩 설명을 붙이고자 할 때 각각의 문단을 장절 체계로 구성해서는 안된다.

사항 해설 번호를 붙여서 사항을 해설해가는 경우 각각의 번호붙임 문단을 장절 체계의 일부라고 생각해서는 안된다.

변형 문단 단순히 문단의 배열 방식을 바꾸어야 하는 경우, 예컨대 예시문이나 인용문은 장절 체계의 일부로 구성해서는 안된다.

장절 표제 문단의 모양만을 사용하기 위한 경우 장절 표제 문단의 모양만을 활용하기 위하여 전후 맥락과는 상관없이 장절 명령을 써서는 안된다.

항목 체계 즉 장절 체계는 문서의 기본 골격이다. [11]에는 차례(장절 체계)가 논문의 논리적 구조를 반영하기 위하여 논문이 작성되는 동안 어떻게 수정되어 가는지에 대한 인상적인 기술이 있다.

8.2 항번

항목 체계가 구성되었다면 이제 여기에 번호붙임을 해야 한다. 항목 체계의 구성이 '글쓰기'의 영역이라면 번호붙임, 즉 항번 체계는 '조판'의 영역에 속한다. 저자 입장에서 첫 번째 큰 항목을 '장'이라고 하든 '절'이라고 하든, 새로운 페이지를 시작하든 그렇지 않든 글의 논리적 구조에 전혀 변함이 없다. 그러나 조판은 글의 내용을 가장 잘 전달할 수 있는 방식으로 번호를 붙이고 문단을 구획하고 페이지 나눔을 결정하고 항번 표제를 시각화해야 한다.

항번 체계의 계층 구조

대표적인 항번 체계를 표 11에 몇 가지 예시하였다. 참고로 예전 정부공문서 규정에는 이 항번 체계를 다음과 같이 정해놓고 있었다.

1, 2, 3, 4 …	(1), (2), (3), (4) …	1), 2), 3), 4) …
가, 나, 다, 라 …	(가), (나), (다), (라) …	가), 나), 다), 라) …

35. 이러한 새로운 명령을 사용자가 정의해서 쓰는 것은 어렵지 않으나 가능하냐의 문제와 바람직하냐의 문제는 다른 것이다.

표 11. 항번 체계의 형식

장절식	숫자-문자식	십진법식
제1부	I. I.	1.
제1장	A. 1.	1.1
제1절	a. (1)	1.1.1
제1관	1. 1)	1.1.2
제1항	(A) 가	1.2
제2장	(a) (가)	1.2.1
제2부		1.2.2

현재는 이 규정이 적용되고 있지 아니하지만 하나의 참고사항으로 활용할 수는 있을 것이다. 아래아한글의 '문단번호' 기능의 기본값은 이것을 토대로 하고 있다. 그러나 생각건대 우리 문서에서 이따금 볼 수 있는 닫는 괄호만을 쓰는 항번은 지양함이 옳지 않은가 한다.

전통적인 우리 문헌의 항번 체계는 부(部)-편(篇)-장(章)-절(節)-관(款)-항(項)-목(目) 순이다. 그러나 책의 장절구분에서 이 순서가 모두 채택되는 예는 극히 드물다. 법조문과 같은 경우 장절 아래 조(條)-항(項)-호(號)의 구획 체계가 쓰이기도 하는데 이것은 장절 항번일 수 없다. 이 글에서는 부와 편을 구분하지 않고 부(편)-장-절까지를 우리말 항번으로 채택한다.

학술문서나 전문과학서 등에서는 십진법식이 주로 쓰인다. 이 방식은 항번의 현재 계층을 명확히 드러낼 뿐 아니라 구조가 한눈에 들어오기 때문에 체계적이라는 장점이 있다. 그러나 번호붙임 자체가 번거롭고 하위 레벨로 내려감에 따라 번호가 너무 길어지면 오히려 식별을 어렵게 할 수도 있으므로 적절한 수준까지만 사용하는 것이 옳을 것이다. memoir의 기본값은 십진법식을 사용하는 것이며 `\subsection`까지만 항번을 붙인다.[36]

최근 출판되는 많은 도서에서는 장절식과 숫자-문자식을 섞어쓰는 추세인 듯하다. 보통 편-장-절까지는 장절식으로, 그 이후로는 십진법식으로 쓰도록 되어 있는데 예컨대 '제 1.1 절'과 같은 것은 이 두 가지가 모두 쓰인 것으로 적절하지 못하다고 생각한다. '제'와 '절'을 붙이려면 '1.1'과 같이 하여서는 아니될 것이다. 그냥 '제 1 절'로 하는 것이 옳다.

항번을 붙이는 것과 관련된 필자의 제안은 다음과 같다.

1. 장절식을 사용하려 한다면 편-장-절까지 장절 명칭을 장절 번호에 붙이고 소절은 간단히 1, 2, 3, ...으로 하는 것을 권한다. 그 아래 수준은 아예 번호를 붙이지 않는다.
2. 그렇지 않고 십진법식을 쓰려 한다면 장에 대해서는 장절식 표제 번호를 붙여도 되겠지만 절부터 '1.1', '1.1.1' 형식을 사용하는 것이 좋다.
3. 숫자-문자식을 사용할 때도 장에 대해서는 장절식 표제 번호를 붙이되, 절부터 숫자-문자식 계층 번호를 붙이는 것이 좋을 것이다.
4. 만약 장이 새로운 페이지에서 시작하지 않는다면 '제'와 '장'을 붙이는 장절식 표제 번호를 쓰지 않을 수 있을 것이다. 이 경우에는 장절 표제 자체가 마치 절 표제처럼

36. memhangul-x에서는 장에 대해서만은 장절식의 표현을 쓰는 것을 기본값으로 하였다.

간략하거나 간소한 형태일수록 좋다.

편장절 번호의 연관성

편은 장보다 상위 계층에 해당하는 최상위 장절 구분 단위이다. 편 대신 부라 하는 책도 많고 드물기는 하지만 편·부를 별개의 항번 계층으로 설정하여 장 앞에 두 수준이 더 있도록 편성하는 책도 있다. 그러나 편이나 부 가운데 하나만을 장보다 상위에 두는 것이 적당하다.

책의 중심적인 흐름은 장이지 편이 아니다. 편은 여러 장들을 편의에 의해 나누어둔 것에 불과하므로 원칙적으로 일종의 표지와 유사한 기능을 하는 것이고 따라서 편이 바뀌었다고 해서 장 번호가 바뀌어서는 안된다. 그러나 책 한 권이 한 가지 내용이 아니라 각 편별로 각각 독립적인 내용으로 이루어져 있고 이것을 묶은 것에 불과한 경우라면 편별로 장 번호를 새로 시작할 수 있다. memoir의 기본 설정에서 장 번호는 편 번호와 무관하게 붙는다. 이것을 바꾸고 싶은 경우라면 내용의 일관성이나 상호 연관성이 없는 경우임을 확인하여야 할 것이다.

절은 장에 종속된다. 따라서 장이 바뀌면 절 번호는 새로 시작한다. 그림이나 표의 번호도 장이 바뀌면 새로 시작하는 것이 합리적이다. 각주 번호는 책 전체에 대해서 일련번호로 붙일 수도 있고 장별로 붙일 수도 있으며 심지어 필요하다면 페이지마다 새로 번호를 매길 수도 있는 것이지만 큰 규모의 책이라면 장별로 각주번호를 매기는 것도 나쁘지 않다.

8.3 장 표제

장이 책의 중심적인 구획이므로 장 표제는 그런 구획이 한눈에 들어오도록 제시되어야 할 것이다. 책에 따라서 장의 구획이 대단히 중요한 것도 있는가 하면 '장'이라고 부르기는 하지만 가벼운 구획 정도에 불과한 것도 있고, 어떤 책은 장별로 아예 다른 내용으로 이루어진 것도 있다. 마지막 경우에는 굳이 장이라 부르지 않고 각각의 글들을 독립해서 취급할 수도 있을 것이다. 그러나 이 경우라 하더라도 구획의 논리적 명칭은 '장'이며 다만 디스플레이 방법이 달라질 따름이다.

장 표제 디자인에 관한 한 일반적인 지침이란 있을 수가 없을 것이라 생각한다. 장 표제를 식자하는 방법은 너무나 많고 이것은 디자이너가 저자의 의도를 가장 잘 전달할 수 있도록 선택하면 그뿐이다. 여기서는 지침을 제시하기보다 실제로 장 표제가 어떤 식으로 구현되어 있는지 몇 가지 사례만을 검토하기로 한다.

별도의 장 표제면

장 표제를 별도의 한 페이지로 구성하는 방법이 있다. 이것은 특히 장 구획이 대단히 중요하고 무게있는 것일 때 가끔 쓰인다. 예를 들어 미술사 책에서 시대별로 각 장을 구성했을 때 아예 새로운 페이지로 새로운 시대에 대한 내용임을 표시하는 경우가 그러하다.

장 표제 페이지를 별도로 설정할 경우 이것은 홀수쪽이 될 수도 있고 짝수쪽이 될 수도 있다. 그런데 홀수쪽을 장 표제 페이지로 만들면 편 표제 페이지와 구별이 어려워지고 만약 편 표제 페이지가 별도로 있는 경우라면 공백면이 너무 많이 생겨난다.

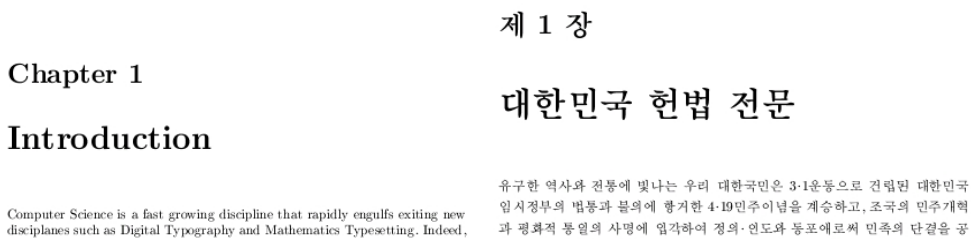


그림 1. 일반적인 장 표제

장 표제 페이지를 짝수면으로 하는 경우 장표제면 삽화를 넣을 수도 있다. 혹은 장 표제 페이지를 홀수면으로 하고 그 내지의 이면인 'verso' 에 장표제면 삽화를 넣는 경우도 있다. 한 페이지를 그냥 표제 문단만 넣고 비우는 것은 너무 심심하기 때문에 장과 관련된 그림을 도판으로 삽입하거나 에피그라프를 넣거나 장내 목차를 별도로 설정하거나 해서 변화를 주는 경우도 있다.

일반적인 장 표제

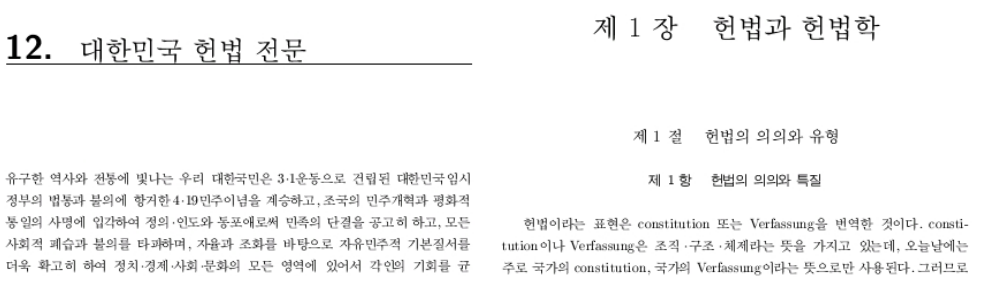
라텍 표준 book, report 클래스의 장 표제는 항번을 본문보다 조금 큰 글자로 찍고 행을 바꾸어서 장 표제 문단을 별행으로 아주 큰 글자로 식자하는 것이다. 장 표제가 시작되기 전에 수직 거리를 일정하게 주고 장 표제 이후에도 비슷한 수직 공백을 설정한다. 그리고 장 표제는 굵은 글꼴로 식자한다. 그림 1의 왼쪽을 참조하라.

memhangul-x에서도 기본값은 이와 거의 같다. 그림 1의 오른쪽을 참조하라. 그러나 필자는 이런 식의 장 표제 식자는 우리 문헌에서 오히려 매우 예외적인 경우에 속한다고 생각한다. 필자는 출판된 책자에서 이런 식의 장 표제를 그대로 사용한 경우를 거의 보지 못하였다. 이 라텍 기본값의 장 표제는 우선 폰트가 지나치게 커서 ──영문의 경우는 적당한 크기인지 모르겠으나── 한글을 식자했을 때는 불안정한 느낌을 줄 뿐 아니라 항번에서 장 표제 문단까지의 거리도 흔히 받아들여지는 것보다 넓다. 반면 장 표제 문단에서 본문까지의 거리는 생각보다 좁다고 느끼는 경우가 많은 듯하다.

몇 가지 장 표제의 예시

책 [10]은 장을 짝수쪽에서 시작한다. 항번은 큰 크기의 굵은 산세리프 숫자로 표기하고 그보다는 한 급 작지만 본문보다는 훨씬 큰 장 표제 문단을 식자한 다음 전체에 대하여 행장 길이의 밑줄을 그었다. 장 표제 상단에는 1행간의 공백을 두었고 장 표제 하단에는 5행간의 공백을 두고 첫 문단을 시작하였다. 장 표제 문단 텍스트는 볼드체로 처리하였다. 그림 2의 왼쪽은 이것을 흉내낸 것이다. 다만 장절 표제 문단의 폰트는 굵은 글꼴로 하지 않았는데 그 이유는 [10]의 장 표제 글꼴이 `\bfseries`에 해당하는 굵은 글꼴이 아니라 기본 글꼴보다 조금 굵은 정도의 태명조 글꼴을 쓰고 있기 때문이다. `\bfseries`를 쓴다면 너무 도드라져 보일 것이다.

우리 문헌에서 자주 만날 수 있는 평범한 것으로는 아마도 책 [1]의 장 표제가 아닐까 한다.



12. 대한민국 헌법 전문

유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에 항거한 4·19민주이념을 계승하고, 조국의 민주개혁과 평화적 통일의 사명에 입각하여 정의·인도와 동포애로써 민족의 단결을 공고히 하고, 모든 사회적 폐습과 불의를 타파하며, 자율과 조화를 바탕으로 자유민주적 기본질서를 더욱 확고히 하여 정치·경제·사회·문화의 모든 영역에 있어서 각인의 기회를 균

제 1 장 헌법과 헌법학

제 1 절 헌법의 의의와 유형

제 1 항 헌법의 의의와 특질

헌법이라는 표현은 constitution 또는 Verfassung을 번역한 것이다. constitution이나 Verfassung은 조직·구조·체제라는 뜻을 가지고 있는데, 오늘날에는 주로 국가의 constitution, 국가의 Verfassung이라는 뜻으로만 사용된다. 그러므로

그림 2. 장 표제의 예시

이 장 표제는 본문과 같은 글꼴을 쓰면서 표제행 상단에도 여분의 공백을 주지 않고 가운데 정렬의 큰 크기의 폰트로 표제 문단을 만든 다음 3행간의 공백을 하단에 넣는 것으로 끝난다. 그림 2의 오른쪽은 이것을 흉내낸 것이다. 이 장 표제는 건조하기 그지없고 변화마저 없어서 법률서적의 특질을 잘 드러내고 있다고 하겠다. 디자이너는 이따금 이렇게 재미없는 선택을 함으로써 내용의 본질을 고스란히 전달하기도 한다.

한 가지 예만 더 들기로 하자. 카라타니 코진의 『일본근대문학의 기원』 (박유하 역, 1997, 민음사)의 각 장은 독립된 글로 다루어져서 별도의 항번을 붙이지 않고 있다. 조금 큰 크기의 폰트로 편집 영역의 상단행에 가운데맞춤으로 장 제목을 식자하고 거의 편집 영역 높이의 반에 이르는 여백을 준 다음에야 본문을 시작하고 있다. 그 결과 장 표제 페이지는 본문이 하단 1/2 정도에만 식자되어 있다는 느낌이 나서 새로운 글이 시작하였음을 시각적으로 알려준다. 장은 홀짝수쪽을 구분하지 않고 시작한다. 이러한 장 표제 문단의 설정은 이 책과 같이 여러 글들을 하나의 책으로 모아놓은 경우에 적합할 것이라 생각된다.

최근 출간되는 비주얼한 책들은 장 표제가 매우 아름답게 치장되어 있는 것이 많다. 그러나 [15]에서 말하는 바 그대로,

"나 좀 봐줘!" 라고 소리지르는 디자인은 타이포그라피 디자인으로서는 최악이다.

라는 교훈을 명심해야 할 것이다. 장 표제의 아름다움만 눈에 들어오고 그 장의 내용이 무엇인지 전혀 기억나지 않는다면 그것은 실패한 디자인이라고밖에 말할 수 없다.

장 표제 디자인의 근간은 단순화이다. 장 표제 디자인이 자연스럽고 평범해야 독자가 표제 디자인을 의식하지 않으면서 내용에 집중할 수 있게 된다. 이것이 너무 요란하면 오히려 역효과를 낼 것이고 너무 무성의하면 그것도 거슬린다.

장 표제의 글꼴

장 표제에는 두 부분이 있다. 하나는 항번이고 하나는 표제 문단이다. 이 둘은 기능적으로 서로 다른 부분에 속하므로 동일한 서체로 식자할 필요는 없다. 그러나 항번과 표제 문단을 별행으로 식자하지 않는 한 동일한 크기를 사용하는 것이 일반적이라고 하겠다. 항번 부분을 별행으로 두는 경우에는 대체로 표제 문단보다 한 단계 작은 서체를 사용할 수 있다.

표 12. memoir 클래스의 기본 글꼴 크기 (본문 10pt 기준)

예시 문장	글꼴 크기 지정 명령	기본값
대한민국	`\tiny`	6pt
대한민국	`\scriptsize`	7pt
대한민국	`\footnotesize`	8pt
대한민국	`\small`	9pt
대한민국	`\normalsize`	10pt
대한민국	`\large`	10.95pt
대한민국	`\Large`	12pt
대한민국	`\LARGE`	14.4pt
대한민국	`\huge`	17.28pt
대한민국	`\Huge`	20.74pt
대한민국	`\HUGE`	24.88pt

종래 본문 10pt 문서에서 장 표제는 20pt 내외의 글자가 주로 쓰였다. memoir 클래스의 `\Huge` 명령(표 12 참조)이 여기에 해당한다. 라텍 표준 클래스에서는 `\Huge`에 해당하는 크기가 20pt가 아니라 24.88pt이다. 이 크기를 장절 표제에서 사용하기 때문에 라텍 표준 클래스의 장절 표제가 지나치게 커보였던 점을 지적해두어야 할 것이다. 아무튼 본문과의 균형을 잃지 않는 정도의 크기를 선택하는 것이 가장 바람직하리라고 생각한다.

장 표제의 서체를 바꿀 것이냐는 것은 전적으로 디자이너의 판단에 속한다. 그러나 장 표제가 큰 크기의 별행으로 식자된다면 굳이 볼드체를 사용할 필요는 없다. 여백 공간만으로도 충분히 제목이 시선을 사로잡기 때문이다. 이 점에서 라텍과 memoir 클래스의 기본값이 볼드체를 기본으로 하고 있는 점은 재고의 여지가 있다 할 것이다. 특별한 디자인 상의 의도가 없다면 어차피 장 표제 자체가 큰 크기의 글꼴을 쓰고 있으므로 이를 또다시 돌출적인 산세리프나 고딕류로 하는 것은 큰 이득이 없다. 본문과 같은 서체로서 폰트 크기와 여백 처리로 장 표제를 배열하는 것이 더 단순하고 이해하기 쉬울 것이라 생각한다.

고딕류를 채택하려 한다면 다음과 같은 점을 고려할 필요가 있다.

- 본문에서는 고딕체가 강조용으로 사용되지 않을 것. 장절 표제에 고딕체가 사용되고 본문에 또다시 강조용으로 사용된다면 번거롭다는 느낌을 피할 수 없을 것이다.
- 장 표제의 폰트 크기가 본문 크기보다 그다지 크지 않을 것.
- 장 표제에서 본문까지 여백 공간이 없거나 1~2 행간 정도만을 둘 것.

장 표제의 자간과 어간

장 표제 문단의 자간과 어간도 주의해서 취급할 필요가 있다. 우선 본문에서는 어떨지 몰라도 장 표제에는 마이너스 자간이 그다지 효과적이지 못하다. 그리고 항번과 표제 문단 사이의

간격이 매우 중요하다. 이 간격이 너무 넓으면 균형감을 잃게 되고 너무 좁으면 장 표제가 답답해보인다.

어간의 경우는 본문 문단의 어간이 어떠하냐에 전적으로 의존한다. 바람직한 것은 본문 문단의 평균적인 어간이 차지하는 자폭에 대한 비율보다 아주 조금 넉넉한 정도가 좋지 않을까 생각한다.

장 표제면의 페이지 스타일

장 표제 자체로 이 페이지는 필요한 정보를 모두 전달하고 있다. 따라서 별도의 면색인은 달지 않아야 한다. 면번호만 있으면 될 것이다.

8.4 편 표제

편 표제면의 디자인은 책 자체의 내용과 조화를 이루어야 한다. 지나친 장식은 피해야 하겠지만 그 자체로서는 독립적인 디자인의 대상이다.

편 표제면이 별면으로 구성되어야 할 이유를 생각해보자. 만약 편(부)의 표제를 만약 장 표제와 같은 면에 둔다면 한 페이지가 시작하자마자 편 표제, 장 표제, 절 표제, 경우에 따라 소절 표제가 한꺼번에 출현하는 페이지가 만들어질 수 있다. 이것은 재앙이다. 편 표제는 앞서 말한대로 일종의 표지(표제면)와 유사한 개념으로서 하나의 독립적 구획이라고 보기 어렵다. 이런 까닭에 편 표제는 별면으로 만드는 것이 일반적이다. 편 표제 별면이 사용되었다면 장 표제는 별면으로 하지 않는다. 편 표제면에는 편 표제를 당연히 넣고 필요하다면 편 목차나 그밖의 요소를 두기도 한다. 편 표제면은 대부분 홀수쪽에 두지만 ──특히 편 목차를 가진다면 홀수쪽에 오는 것이 좋다── 짝수쪽에 두는 경우도 없지 않다. 그런데 짝수쪽에 편 표제면을 두었을 때 그 건너편 쪽인 홀수쪽에서 장이 시작한다면 편 표제면이 두드러지는 것을 피할 수 없고 이것은 그다지 좋은 선택이 아닐 것이다.

일반화해서 말하자면 장 표제가 짝수쪽에 온다면 편 표제도 짝수쪽에 올 수 있겠지만 장 표제가 홀수쪽에 온다면 편 표제도 홀수쪽에 두고 그 이면을 비우는 것이 좋으리라 생각한다. 이 경우 편 표제면은 마치 끼워넣은 한 장의 내지인 것처럼 취급될 수 있다. 그러나 편 표제면은 페이지 번호매김이 이루어지며 면번호를 장 표제와 같이 페이지 하단에 표시하는 경우도 많다. 독자가 편의 위치를 쉽게 찾을 수 있도록 페이지 바깥까지 벗어나는 탭을 붙여두는 경우도 있다. 혹은 편 안의 모든 오른쪽 페이지 끝에 고정된 위치의 탭을 붙여두기도 한다. 그러나 이것은 사전과 같이 위치 색인이 필수적인 검색 도구가 되는 경우가 아니라면 두지 않는 것이 일반적이다.

8.5 절 표제

장 표제와는 달리 절 표제에는 고려해야 할 점이 몇 가지 있다.

동행 표제와 별행 표제 표제를 별행으로 식자하고 본문을 표제 아래 별도 문단으로 작성할 것인지 아니면 표제를 문단의 일부처럼 식자할 것인지 결정해야 한다. 일반적인 문서에서

\section은 대부분 별행 표제로 작성된다. 라텍에서는 \paragraph부터 동행 표제가 되는 것이 기본값이다.

들여밀기 별행 표제로 하는 경우 절 표제의 들여밀기를 어느 정도로 할 것인지를 결정해야 한다. 보통 절 표제의 경우에는 변이단 조판이 아니라면 본문 문단과 왼끝을 가지런하게 하여 작성하는데 경우에 따라 조금 들여밀기를 하거나 항번 영역을 여백으로 끌어내는 경우도 있을 수 있다.

폰트 절 표제는 장 표제보다는 작은 크기의 폰트를 쓰는 것이 일반적이다. 라텍에서는 절 표제에 일관되게 굵은 글꼴을 쓰고 있지만 이것은 반드시 요구되는 사항이 아님을 앞서 지적하였다. 절 표제나 그 하위 항목의 폰트를 산세리프 등으로 바꾸는 것은 무책임하게 할 경우 판면을 혼란스럽게만 할 우려가 있다. 폰트에 대한 확신이 없다면 본문 폰트를 크기만 변경하는 것이 안전하다.

항번 붙이기 경우에 따라 하위 절 표제는 항번을 붙이지 않고 표제 문단만 남기는 경우가 있다. 일반적으로 말해서 절이나 소절까지는 항번을 붙이는 것이 구획을 식별하기 좋고 그보다 하위 구획 단위에 대해서는 할 수만 있다면 항번을 붙이지 않아도 될 경우 붙이지 않는 것이 좋다. 너무 깊은 깊이의 항번 체계는 독자의 현재 위치 파악을 언제나 어렵게 만든다. 항번을 붙이지 않는 하위 절 표제는 폰트를 치환하거나 하여 그 깊이를 한눈에 알아볼 수 있게 하여야 할 것이다.

숫자 항번 다음의 점 숫자만으로 이루어진 항번에 점을 찍을 것인가? 즉 절 항번이 '1' 이라면 이것을 '1.' 으로 표시해야 하느냐의 문제가 있다. 라텍에서는 일반적으로 절 항번 다음에 점을 찍지 않는다. 특히 '1.1' 과 같은 형식의 절 항번 다음에는 점을 찍어서 '1.1.' 이라 하면 어색하다. 절 항번과 절 표제 문단 사이에는 약간의 간격을 두게 되므로 점을 찍지 않는 것이 좋을 것으로 생각한다. 그러나 만약 절 항번이 문자로 이루어져 있다면 점을 찍는 것도 나쁘지 않은 선택이 될 것이다. 예컨대 이 문서의 경우처럼 '가.' 과 같이 절 항번을 표시하는데 여기에 점을 찍지 않으면 표제 문단과 혼동을 일으킬 수 있다.

절 표제가 너무 두드러지면 글읽기의 흐름을 깨뜨리고 너무 본문에 묻혀버리면 장절 구획을 식별하기 어려워진다. 아름답고 논리정연한 절 구획과 절 표제 디자인은 쉽지 않은 문제인만큼 중요하다고 하겠다.

8.6 첫 단락 들여밀기

라텍 표준 클래스의 기본값으로 작성한 문서에서 자주 만나는 질문 중에 하나가 첫 단락이 들여밀기되지 않는다는 불평이다. indentfirst 패키지를 이용하면 첫 단락을 들여밀기할 수 있다고 답변하기는 하는데, 이것이 '불평' 으로 나타나는 이유는 아마도 우리가 어린 시절부

터 대하는 교과서를 위시한 많은 도서들이 첫 단락 들여밀기를 하고 있는 데서 비롯된 습관 때문이 아닐까 한다.[37]

그런데 가만 생각해보면 '들여밀기'란 문단 구획에 지나지 않는다. 첫 단락은 이미 장절 표제 아래 있다는 사실 자체가 하나의 문단이 시작되는 위치임을 직관적으로 확인할 수 있는데 여기서 또하나의 문단 구획이 필요할 이유가 무엇인가? 게다가 장절 표제가 만약 중앙 정렬로 식자되는 경우라면 첫 단락 들여밀기는 필연적으로 판면을 기우뚱하게 만든다. 따라서 되도록이면 첫 단락 들여밀기를 하지 않는 것이 올바른 판면 구성방법이 될 것이라고 필자는 생각하고 있다. 그러나 디자이너가 첫 단락을 들여밀어서 어떤 효과를 얻으려 한다면 그렇게 할 수 있을 것이다.

9 장절 표제 만들기

이제 memoir와 memhangul-x로 문서를 작성할 때 장절 표제를 만드는 문제를 취급하기로 하자. memoir 클래스는 '장 스타일'이라는 개념을 포함하여 장 표제를 제어할 수 있는 다양한 도구를 제공한다. memhangul-x는 한글 문서를 위한 추가 매크로 몇 가지를 제공하고 있다.

9.1 장 스타일

표 13에 memoir와 memhangul-x의 장 스타일 일부를 요약하였다. 이외에도 상당히 많은 장 스타일들이 기본으로 제공되고 있으므로 [14]를 참고하라. memhangul-x는 이 대부분의 장 스타일을 '제 1 장' 형식으로 식자하도록 수정해두고 있다.

장 스타일 '`appendix...`'은 memhangul-x에서 새로이 정의한 것이다. 이것이 필요했던 이유는 우리 문헌의 부록 장 표제 식자 방식이 본문의 장 표제 식자 방식과 많이 달라서 사실상 동일한 장 스타일을 유지할 수 없었던 데 있다. 물론 부록 범위 안이라면 특정한 장 스타일이 되도록 조건문을 이용하여 정의할 수도 있겠지만 부록이 문헌의 필수 구성부분이라 보기는 어려웠기 때문에 위와 같이 새로운 장 스타일을 추가하는 것으로 대신하였다. 경제성을 생각한 판단이었지만 조금 번거로워진 것은 어쩔 수 없다.

memoir 클래스에 정의되어 있는 장 스타일을 그대로 한글 문서에 쓰면 약간의 어긋남이 발생한다. 필자가 작성한 ob-chapstyles 패키지는 memoir의 기본 정의를 한글화한 것이다.

장 스타일 설정

일견 memoir의 장 스타일 설정 방법은 사용자의 의도를 장 표제 스타일로 구현할 수 있다는 점에서는 대단히 편리하지만 약간 많은 양의 매크로를 설정하거나 정의해야 한다는 부담스러움은 분명히 있다. 그렇기 때문에 범용으로 쓰일 만한 장 스타일들을 공개하여 공유하기를 희망한다. 자신이 작성한 장 스타일을 공개하여둔다면 그것을 바탕으로 더 좋은 스타일들이 개발되고 이용할 수 있게 될 것이다.

37. 여담이지만 이른바 '교과서'는 조판의 관점에서 보면 허술한 데가 한두 군데가 아니었다. 장절 편성의 비체계성, 페이지 여백의 협소함, 폰트의 (아니) 미려함, 수식 조판의 허술함 등은 자주 지적되는 문제들이다.

표 13. memoir와 memhangul-x의 장 스타일

장 스타일	설명
default	라텍의 기본 스타일과 동일하다. 기본값.
section	\chapter를 \section처럼 식자하는 스타일이다. 번호만이 표시되고 장 표제 문단과 항번은 같은 줄에 놓인다.
hangnum	section 스타일과 유사하지만 항번이 왼쪽 마진 영역까지 끌어내어진다.
companion	책 [12]의 장 표제를 흉내내는 것이다. 항번과 장 표제가 오른쪽 정렬되면서 오른쪽 여백을 침범한다.
article	라텍 article 클래스의 \section과 유사한 모양으로 장 표제를 식자한다. section 스타일과 유사하지만 폰트와 간격 설정이 다르다.
demo	장 표제의 아래 위로 선을 긋고 산세리프로 장 표제를 식자한다.
demovar	demo 스타일과 같지만 항번 표시 방식이 '일 장'과 같이 표기 된다.
appendixdefault	부록에서 쓰기 위한 default 스타일이다.
appendixsection	부록용 section 스타일이다.
appendixcompanion	부록용 companion 스타일이다.
appendixdemo	부록용 demo 스타일이다.

장 스타일의 길이 변수들은 다음과 같다.

\beforechapskip	장 표제의 앞에서 띄우는 수직 거리이다. default 스타일에서 50pt.
\midchapskip	장 표제 이후의 수직 거리값이다. default 스타일에서 40pt.
\afterchapskip	장 표제 내의 항번과 표제 문단 사이 수직 거리이다. 기본값은 20pt.

앞서 논의한 대로 장 표제 식자에는 본문 폰트와 다른 두 종류의 폰트를 쓰게 된다. 원한다면 항번 폰트와 표제 문단 폰트를 달리 할 수 있어야 하는 것이다. 그리고 숫자 자체를 위한 폰트를 별도로 선택할 수도 있다. 장 표제 식자를 위한 폰트 선택 관련 매크로는 다음과 같다.

\chapnamefont	장 이름을 식자하는 데 사용하는 폰트. 장 이름은 대부분의 경우 '장'.
\chapnumfont	장 번호 숫자를 식자하는 데 사용하는 폰트.
\chaptitlefont	장 표제 문단을 식자하는 데 사용하는 폰트.

memhangul-x의 장 표제 출력 루틴은 다음과 같다. memoir와 달리 \printchaptername은 사용되지 않는다.

```
\chapterheadstart
\memucsinterwordchapterskiphook
  \prechapternum
  \chapternamenum
  \printchapternum
  \chapternamenum
  \postchapternum
  \afterchapternum
  \printchaptertitle{장 표제 문단의 내용}
  \afterchaptertitle
```

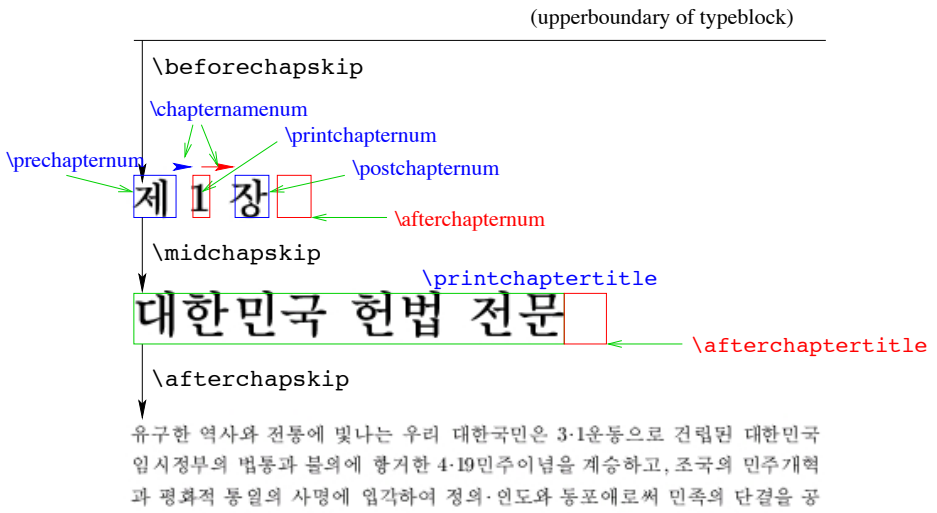


그림 3. 장 표제 파라미터

`\frontmatter`나 `\backmatter`에 오는 '번호붙임 없는' 장 표제를 식자할 때는 다음 루틴을 따른다.

```
\chapterheadstart
\memucsinterwordchapterskiphook
  \printchapternonum
  \printchaptertitle{번호 붙지 않은 장 표제}
  \afterchaptertitle
```

우선 논의의 번거로움을 피하기 위해 번호붙임 없는 장 표제를 잠시 뒤로 미루고 일반적인 장 표제 스타일 설정을 이해해보기로 하자. 라텍 표준 스타일과 크게 다를 것이 없는 `default` 스타일의 디자인이다.

그림 3은 위의 의사 코드(pseudo-code)가 실제로 어떻게 구현되고 있는지를 보여준다. 이것을 구현하는 방법을 생각해보자.

```
\makechaptertitle{mydefault}{%
```

이 장 스타일의 이름을 임시로 '`mydefault`'라고 하기로 한다. 모든 장 스타일 명령들은 이 매크로의 두 번째 인자 안에서 이루어진다.

```
  \setlength{\beforechapskip}{50pt}
  \setlength{\midchapskip}{20pt}
  \setlength{\afterchapskip}{40pt}
  \renewcommand{\chapterheadstert}{\vspace*{\beforechapskip}}
```

길이값의 지정이다. 위에 설정된 값은 라텍 표준 book 클래스의 기본값이다. 여기에서 `\chapterheadstart`는 별도로 정의하지 않아도 된다. 그러나 이것이 어떻게 정의되는지를 보이기 위하여 재정의해보았다.

```
6     \renewcommand{\chapnamefont}{\normalfont\huge\bfseries}
7     \renewcommand{\chapnumfont}{\normalfont\huge\bfseries}
8     \renewcommand{\prechapternum}{\chapternamefont 제}
9     \renewcommand{\postchapternum}{\chapternamefont 장}
10    \renewcommand{\chapternamenum}{\space}
11    \renewcommand{\printchapternum}{\chapnumfont\thechapter}
12    \renewcommand{\afterchapernum}{\par\nobreak\vskip\midchapskip}
```

당연히 \thechapter는 예컨대 \arabic{chapter}와 같이 미리 정의되어 있어야 한다. 11번 줄은 항번을 식자하는 코드이다. 장의 이름인 '장'은 \@chapapp에 이미 정의되어 있으므로 그것을 사용해도 되고 memhangul-x에서는 \pre@chapter가 '제'로 \post@chapter가 '장'으로 미리 정의되어 있으므로 이것을 이용해도 된다. 항번 이후에 개행하고 \midchapskip 값을 수직간격으로 주는 것이 \afterchapnum 매크로에 할당되어 있다.

```
13    \renewcommand{\chaptitlefont}{\normalfont\Huge\bfseries}
14    \renewcommand{\printchaptertitle}[1]{\chaptitlefont ##1}
15    \renewcommand{\afterchapertitle}{\par\nobreak\vskip\afterchapskip}
```

장 표제 문단을 식자하는 코드이다.

```
16    \renewcommand{\printchapternonum}{}
17    \renewcommand{\hchaptertitlehead}{제~\thechapter~장}
18  }
```

\printchapternonum은 번호붙임 없는 장 표제를 식자할 때 장 표제 문단 이전에 오는 코드를 넣는 곳이다. 여기서는 단순히 비워둔다. 즉 번호붙임 없는 장 표제를 식자할 때는 장 표제 문단을 쓰기 전에 아무 것도 하지 말라는 뜻이다. \hchaptertitlehead는 memhangul-x 매크로인데 목차와 면주에 장 표제를 적을 때 항번 형식 '제 1 장'을 장 표제 행 앞에 붙이도록 하는 설정이다. 원한다면 목차와 면주에서는 '1.'과 같이 쓸 수도 있는 것인데 그럴 경우에는 위의 정의를 수정하여야 한다. 이 매크로는 면주 설정 매크로에서 호출되어야 할 것이다. 마지막 18번 줄의 닫는 괄호는 mydefault 장 스타일 정의가 끝났음을 표시한다.

장 스타일 정의에 절 또는 소절의 형식도 함께 정의할 수 있다. 절 표제 설정 명령도 함께 적어주면 \chapterstyle{mydefault}라고 하였을 때 그 장의 섹션 등에서 효력을 발생할 것이다. 다음과 같이 문서에서 정의하면 이 장 스타일이 적용된다.

```
\chapterstyle{mydefault}
```

이밖에 장 스타일과 관련된 매크로를 정리하자. \clearforchapter는 장 시작 페이지를 만드는 매크로이다. 내부적으로 이용되고 있으므로 사용자가 별도로 지정해주어야 할 필요는 없다. \openright의 경우 \cleartorecto, \openleft의 경우 \cleartoverso, \openany의 경우 \clearpage에 각각 할당되어 있다. 표 2를 보라.

\insertchapterspace는 그림이나 표 목차에서 장이 바뀔 때 약간의 수직 간격을 넣어서 구분해준다. 역시 별도로 부를 필요는 없다. 다만 만약 장이 바뀔 때에도 공백을 넣고 싶지 않다면 이 매크로를 재정의해서 아무 일도 하지 않도록 바꾸면 된다.

장 스타일 디자인과 구현

이제 위의 장 스타일 설정 방법을 이용하여 그림 2에서 보인 장 표제를 구현해보자.

먼저 장 번호 숫자를 표기하는 폰트를 28pt 산세리프 계열 숫자로 할 것이다. 여기에 해당하는 기정값 폰트 지시 매크로가 없으므로 이것을 정의하자.

```
1  \def\TITLENUMFONT{\normalfont\sffamily\bfseries\fontsize{28}{30}\selectfont}
```

이 장 스타일의 명칭을 `leejw`이라고 부르기로 하고 정의를 시작한다.

```
2  \makechapterstyle{leejw}{%
```

장 표제 상단에는 1행간을 넣는다. 하단에는 5행간을 넣는다. 우리는 1.333 행간을 쓰고 있으므로 `\onelineskip`에 이 값을 곱해주어야 한다.

```
3      \setlength{\beforechapskip}{1.333\onelineskip}
4      \setlength{\afterchapskip}{6.67\onelineskip}
5      \setlength{\midchapskip}{0pt}
```

몇 가지 기본값 간격 매크로를 정의한다. 항번과 장 표제 문단은 동일한 선상에 있으므로 개행과 수직 간격을 주지 않는다.

```
6      \renewcommand{\chapterheadstart}{\vspace*{\beforechapskip}}
7      \renewcommand{\afterchapternum}{\hskip18pt}
```

폰트에 관련된 설정을 행한다. `\Huge`는 약 20pt이므로 장 표제 문단 식자에 적절하다.

```
8      \renewcommand{\chapnamefont}{\TITLENUMFONT}
9      \renewcommand{\chapnumfont}{\TITLENUMFONT}
10     \renewcommand{\chaptitlefont}{\normalfont\bfseries\Huge}
```

이제 항번 부분을 설정한다.

```
11     \renewcommand{\chapternamenum}{}
12     \renewcommand{\prechapternum}{}
13     \renewcommand{\postchapternum}{}
14     \renewcommand{\printchapternum}{\chapnamefont\thechapter.}
```

장 표제 문단을 식자하는 코드를 추가한다.

```
15     \renewcommand{\printchaptertitle}[1]{\chaptitlefont ##1}
16     \renewcommand{\afterchaptertitle}{%
17         \par\nobreak\vskip2pt\hrule\vskip\afterchapskip}
18 }
```

위의 코드에서 `\afterchaptertitle`은 `\textwidth` 길이의 가로선을 하나 그리고 난 후 `\afterchapskip`을 추가하는 것이다.

이 스타일에는 한 가지 문제점이 있다. 그것은 장 표제가 한 행을 넘치는 경우 괘선이 그어지는 위치가 항번에 대하여 부적절하다는 것이다. 원래의 책에서는 그런 경우가 전혀 없기 때문에 문제가 되지 않았지만 경우에 따라 그럴 경우의 에러처리도 해주면 좋을 것이다. 그러나 여기서는 그러한 문제점을 지적하는 것으로 만족하기로 한다.

\chapter 명령

이제 문서의 본문에서 \chapter 명령을 부르는 방법을 알아본다. 가장 간단한 것은 다음과 같이 한 개의 인자로 장 표제 문단 텍스트를 지시하는 것이다.

```
\chapter{대한민국 헌법 전문}
```

별표 붙은 \chapter 명령은 장 표제 문단만을 식자하고 면주를 수정하지 않고 목차에 넣지도 않는다. 그리고 항번도 증가하지 않는다.[38]

```
\chapter*{번호붙임 없는 장 표제}
```

\chapter 명령은 두 개의 옵션 인자를 가질 수 있는데 만약 하나의 옵션 인자만이 주어지면 목차와 면주에 들어갈 텍스트로 옵션 인자를 사용한다. 두 개가 주어지면 첫 번째 옵션 인자가 목차에, 두 번째 옵션 인자가 면주에 사용된다. 옵션 인자는 표제 문단 텍스트가 특히 길어서 면주 등에 넣기에 적당하지 않을 때 사용한다.

```
\chapter[목차 표제행][면주 표제행]{표제 문단 텍스트} % 또는
\chapter[목차/면주 표제행]{표제 문단 텍스트}
```

별표 붙은 명령은 한 개의 옵션 인자만을 가질 수 있다. 만약 별표 붙은 명령에 옵션 인자가 사용되면 면주에 이것이 나타난다.

```
\chapter*[면주 표제행]{표제 문단 텍스트}
```

\chapter 명령을 부르는 것은 항상 \thispagestyle{chapter}를 부르는 효과를 가진다. 페이지 스타일 chapter의 기본값은 plain 스타일과 같다. 즉 페이지 바닥의 중앙에 페이지 번호만을 보통 폰트로 찍어준다. 이 설정이 마음에 들지 않는다면 페이지 스타일 chapter를 수정하면 된다.

9.2 편 표제 만들기

편 표제는 비록 '편 스타일'이라는 것을 별도로 제공하지는 않지만 장 스타일 정의 매크로와 유사한 방법으로 스타일을 설정할 수 있다. 편 표제는 항상 새로운 페이지로 시작하고 part 페이지 스타일을 적용한다. 표 14는 편 표제를 식자하기 위하여 고려할 수 있는 제어 문자열들을 정리한 것이다.

memoir 클래스와는 달리 memhangul-x는 \printpartname을 사용하지 않는다. 그리고 \hparttitlehead를 별도로 가지고 있다. 항번을 위해 \prepartnum이나 \postpartnum과 같은 매크로가 사용된다. 장 표제의 경우와 달리 \beforepartskip 등의 ...skip 명령이 길이값이 아니라 명령이므로 \setlength로 설정하는 것이 아니라 \renewcommand로 재정의해주어야 한다. 다음은 예제이다.

```
\renewcommand{\beforepartskip}{\null\vfil}
\renewcommand{\afterpartskip}{\vfil\clearpage}
\renewcommand{\midpartskip}{\par\vskip20pt}
```

\parttitlefont는 부록 조판시의 \appendixpage에도 사용된다.

38. 장 표제의 항번을 증가시키고 목차나 면주에도 표시하면서 항번 자체를 인쇄만 하지 않으려면 §9.3을 보라.

표 14. 편 표제 설정 매크로

매크로	설명
\beforepartskip	편 표제 앞에 둘 수직 간격 명령이다.
\afterpartskip	편 표제 뒤에 둘 수직 간격 명령이다.
\midpartskip	편 표제 항번과 표제 문단 사이의 길이값 설정 명령이다.
\partnamefont	편 이름 식자에 사용할 폰트를 지시한다.
\partnumfont	편 항번 숫자에 사용할 폰트를 지시한다.
\parttitlefont	편 표제 문단 텍스트에 적용할 폰트를 지시한다.
\prepartnum	항번 앞 수식어. (‘제’)
\postpartnum	항번 뒤 수식어. 보통 편 이름이다. (‘편’)
\partnamenum	항번 앞뒤 수식어와 항번 숫자 사이의 간격이다.
\printpartnum	편 항번 숫자를 식자한다.
\printparttitle	편 표제를 식자한다.
\hparttitlehead	편 표제 항번을 면주와 목차에 표시하기 위한 설정이다.

표 15. 절 표제 설정 매크로

매크로	인자	설명
\setbeforeSECskip	skip	절 표제 시작 전 수직 공백 크기이다.
\setSECindent	len	절 항번과 표제의 들여쓰기 값이다.
\setSECheadstyle	text	절 항번과 표제의 스타일과 폰트를 지정한다.
\setafterSECskip	skip	절 표제 뒷쪽 공백 크기를 나타낸다.
\sethangfrom	code	\@hangfrom 매크로를 재설정한다.
\setsecnumformat	code	절 항번 형식을 지정하는 \@seccntformat 매크로를 정의한다.
\hangsecnum		절 항번 부분만 왼쪽 여백으로 내어밀기된다.
\defaultsecnum		\hangsecnum 설정을 되돌린다.
\setSEChook	text	\SEChook 매크로는 절 표제를 식자하기 직전에 호출되는 매크로이다. 이 매크로를 재설정한다.

9.3 절 표제 만들기

편이나 장과는 달리 절, 소절 등 ‘하위 수준 장절명령’에 대해서 memhangul-x는 별다른 조치를 취해두고 있지 않다. 따라서 memoir 클래스의 설정 방법을 그대로 이용하면 된다.

memoir 매뉴얼 [14]의 예와 마찬가지로 SEC이라는 부호를 사용하겠다. 이것은 sec, subsec, subsubsec, para, subpara가 올 수 있는 약어이고 각각

```
\section, \subsection, \subsubsection, \paragraph, \subparagraph
```

를 나타낸다. 말하자면 \setbeforeSECskip이라는 매크로는 각각 \setbeforesecskip 또는 \setbeforesubsecskip 등을 나타낸다. 표 15는 절 표제를 설정하기 위해 사용되는 매크로를 요약한 것이다.

절 표제를 설정하려 할 때 부딪히는 문제 몇 가지를 사례별로 짚어보겠다.

표 16. \beforeSECskip 및 \afterSECskip 기본값

절 표제 수준	\beforeSECskip 기본값	\afterSECskip 기본값
section (sec)	−3.5ex plus −1ex minus −.2ex	2.3ex plut .2ex
subsection (subsec)	−3.25ex plus −1ex minus −.2ex	1.5ex plus .2ex
subsubsection (subsubsec)	−3.25ex plus −1ex minus −.2ex	1.5ex plus .2ex
paragraph (para)	3.25ex plus 1ex minus .2ex	−1em
subparagraph (subpara)	3.25ex plus 1ex minus .2ex	−1em

절 표제의 상하 간격 조절

라텍은 절 표제를 만들 때 내부 명령인 \@startsection을 부른다. \@startsection은 모두 여섯 개의 인자를 취하는데 각각 다음과 같다.

- #1 장절의 카운터 명칭. \section에 해당하는 카운터는 'section' 이다.
- #2 장절의 레벨. \section의 레벨은 1이다.
- #3 장절 표제의 들여쓰기. \section에서 \paragraph까지는 0pt, \subparagraph는 \parindent이다.
- #4 장절 표제 앞쪽의 간격 \beforesecskip.
- #5 장절 표제 뒷쪽의 간격 \aftersecskip.
- #6 장절 표제 식자 직전에 오는 명령. 주로 장절 표제의 폰트와 정렬 방식을 지시한다.

memoir 클래스는 이 각각의 인자들을 제어하는 매크로들을 별도로 제공하고 있다. 표 15를 참조하라.

예컨대 절 표제의 상하 간격을 조절하려면 \setbeforeSECskip 매크로를 이용해 값을 지정해주면 된다. 이 매크로의 기본값은 표 16에 있다. 여기서 \section의 경우는 −3.5ex plus −1ex minus −.2ex로 되어 있으므로 —이 값이 음수인 것은 첫 단락 들여밀기와 관련되므로 다음 항목 '절 표제 다음 첫 단락 들여밀기' 를 참고하라— 실제 앞 단락 마지막과 절 표제 사이의 거리는 \beforesecskip + \parskip + \baselineskip 값에 해당할 것이다.

절 표제에서 첫 단락까지의 간격은 \setafterSECskip 매크로로 설정한다. 이 매크로의 기본값도 표 16에 있다. 표제 \section의 경우 실제 간격은 \aftersecskip + \parskip + \baselineskip에 해당하는 값이 될 것이다.

절 표제 다음 첫 단락 들여밀기

표 16에 보인 \beforeSECskip의 값이 음수이면 다음 첫 단락이 들여밀기되지 않는다. 반면 양수이면 첫 단락 들여밀기로 조판한다. 기본값이 음수로 되어 있다. 표에서 알 수 있듯이 \section, \subsection, \subsubsection은 음수이고 \paragraph와 \subparagraph는 양수이다.

이 값을 다음과 같이 양수로 변경하면 절 다음 첫 단락을 들여밀기할 수 있다.

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

조금 다른 문제이기는 하지만 장 다음 첫 단락 들여밀기를 위해서는 이것과는 다른 방법을 써야 한다. 편과 장을 식자하는 코드를 유심히 살펴보면 `\@afterheading`이라는 명령이 있다. 이 매크로는 내부적으로 `\if@afterindent`라는 조건문을 사용하여 첫 단락 들여밀기 여부를 식자에 이용하고 있다. 그러므로 `\@afterheading`이 불릴 때 이 조건의 값을 `true`로 해주면 첫 단락 들여밀기가 될 것이다. memhangul-x는 이를 위한 매크로를 하나 제공한다.

```
\chapterindentfirst
```

이것을 되돌리는 명령은 제공되지 않으므로 프리앰블에서만 사용하도록 함이 좋을 듯하다. 장의 경우에는 이밖에도 에피그라프가 있을 경우에는 첫 단락 들여밀기가 무력화되어 첫 단락이 들여밀기되기 때문에 오히려 첫 단락을 들여밀기하지 않으려면 `\noindent`를 써주어야 할 때가 있다.

동행 표제로 절 표제 식자

동행 표제란 절 표제가 문단 첫머리에 놓이고 문단 본문이 같은 행에서 시작하도록 설정하는 것을 말한다. `\afterSECskip` 값이 음수일 때 동행 표제가 된다.

`\afterSECskip` 값이 양수일 때 이 간격값은 수직 간격을 의미한다. 반면 음수일 때는 그 절대값이 표제 마지막과 문단 첫 글자 사이의 수평 간격을 의미한다. 예를 들어 표 16에서 `\paragraph`는 이 값을 음수로 정의하고 있는데 기본값은 −1em이다. 이것은 `\paragraph` 표제 다음에 1em의 수평 간격을 두고 문단이 시작한다는 뜻이다.

절 항번을 '제 1 절' 형식으로 바꾸기

우선 `\thesection`을 재정의한다. `\thesubsection`도 함께 정의해보자.

```
\renewcommand\thesection{제~\arabic{section}~절}
\renewcommand\thesubsection{\arabic{section}.\arabic{subsection}}
```

그런 다음에 이 형식을 `\setsecnumformat`으로 설정한다.

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

위의 명령은 `\section`뿐 아니라 `\subsection` 등이 불릴 때도 동일한 모양으로 설정된다. 즉 `\section`의 경우는 '`\thesection\quad`'로 구현되는 반면, `\subsection`의 경우는 '`\thesubsection\quad`'로 구현될 것이다.

만약 `\subsection`은 이와 다른 형식을 쓸 생각이라면 ──아래 예제에서는 끝에 점을 하나 더 찍어보았다── 위의 설정이 오직 `\section`에만 미치도록 해야 한다. 그러므로 `\sechook` 매크로를 이용하여 `\section`이 불릴 때 이 코드를 삽입하도록 다음과 같이 한다.

```
\setsechook{\setsecnumformat{\thesection\quad}}
\setsubsechook{\setsecnumformat{\thesubsection.\quad}}
```

위의 예는 특별히 `\section`과 `\subsection`의 항번과 표제 식자 양식이 다른 경우를 예로 든 것이다.

이 정의에서 한 가지 주의할 점이 있다. \setsecnumformat 매크로를 \setsechook 등의 인자로 쓰거나 장 스타일 정의(\makechapterstyle) 안에서 정의하려면 #1과 같은 인자 매크로 문자를 사용할 수 없다. 그래서 위와 같이 \thesection을 써야 하는 것이다. \thesection을 쓰게 되면 그 이후의 모든 절류 명령들이 \thesection을 기준으로 만들어진 표제 형식을 갖게 된다. 그러므로 sec과 subsec 등의 항번 스타일을 달리 하고자 한다면 반드시 필요한 하위 수준 절 표제 형식을 각각 \setSEChook을 통하여 정의해 주어야 한다.

처음에 설명한 것과 같이 동일한 항번 형식을 취하면 \thesection 같은 명령들을 재정의하는 것만으로 충분하므로 번거로움을 줄이는 방법이 될 것이다.

절 항번을 1.1 형식으로 바꾸기

위의 방법을 그대로 응용하여 다음과 같이 한다.

```
\renewcommand\thesection{\arabic{chapter}.\arabic{section}}
```

절 항번 뒤에 ':'를 붙이고 표제 텍스트까지 간격 두기

\setsecnumformat 매크로를 이용하여 다음과 같이 정의한다.

```
\setsecnumformat{\csname the#1\endcsname :\hskip 1em}
```

여기서 '\hskip 1em' 대신 다른 간격 명령 \quad 등을 사용하는 것도 좋다. \section에서만 효과가 있게 하고 \subsection은 다른 방법으로 정의하고자 한다면 앞에서 설명한 바와 같이 \sechook 또는 \subsechook을 다음과 같이 재정의하면 된다.

```
\setsechook{\setsecnumformat{\thesection :\hskip 1em}}
\setsubsechook{\setsecnumformat{(\thesubsection )\hskip 1em}}
```

절 표제의 폰트 바꾸기

\section의 표제 폰트 기본값은 '\Large\bfseries\raggedright'이다.

```
\setsecheadstyle{\normalfont\sffamily\Large\bfseries\centering}
```

이 매크로는 \secheadstyle이라는 매크로를 재정의하는 역할을 하는데 폰트만이 아니라 정렬 방법도 함께 지정한다. 위의 예는 가운데 정렬하도록 설정한 경우이다. 정렬 명령을 맨 마지막에 둔다. 보다 하위 명령인 \setsubsecheadstyle과 \setsubsubsecheadstyle 등도 같은 방법으로 사용하면 된다.

이 매크로의 재미있는 점은 마지막에 오는 명령은 한 개의 인자를 취할 수도 있다는 것이다. 그러므로 예를 들어서

```
\setsecheadstyle{\normalfont\sffamily\Large\textbf}
```

과 같이 설정하여도 마지막의 \textbf가 한 개의 인자를 안전하게 취하여 원하는 결과를 얻을 수 있게 된다.

소절 이후로 항번 붙이지 않기

\subsection 이후로 항번을 붙이지 않지만 목차에는 나타나게 하려면 \subsection*과 같은 별표붙은 명령을 써서는 안된다. 번거롭게 \addtocontentsline를 한 줄 더 쓰는 방법이 있기는 하지만 그리 권장할 만하지 못하다. 이럴 때 편리하게 쓸 수 있는 명령이 memoir에 있는 매크로 \setsecnumdepth이다. 예를 들어 \subsection까지만 항번을 붙이려면

```
\setsecnumdepth{subsection}
```

으로 한다. \maxsecnumdepth는 번호붙임의 수준의 최대값을 지정하는데 \mainmatter 명령이 불릴 때 \secnumdepth를 \maxsecnumdepth로 설정한다. memoir 기본값은 이 두 값이 'section'으로 설정되어 있다. 즉 memoir 기본값 문서에서는 \subsection부터 항번이 붙지 않는다.

절 표제 들여밀기

이따금 하위 절 표제를 \indent 만큼 들여밀기하고 싶을 때가 있다. \setSECheadstyle의 마지막 인자로 \indent를 주어도 들여밀기는 될 것이다. 그러나 \setSECindent라는 별도의 매크로도 있다. 이 두 가지는 무엇이 다른가? 예를 들어 \subsubsection을 \indent 만큼 들여밀기하려 한다고 하자.

```
\setsubsubsecheadstyle{\normalfont\large\bfseries\indent}
```

이렇게 설정한 경우라면 \subsubsection의 표제 문단 텍스트가 두 줄 이상이 될 때 두 번째 줄부터 들여밀기되는 크기는 \noindent한 경우의 항번 표제 길이만큼일 것이다. 그러나 다음과 같이 한 경우는 항번 부분만큼이 내어밀기되어 항번 표제 문단이 가지런하게 정렬된다.

```
\setsubsubsecindent{\parindent}
```

다시 말하면 이럴 경우 되도록 \setSECindent를 사용하는 것이 옳다고 하겠다. 설령 두 줄 이상이 되는 \subsubsection이 없다 하더라도 그렇다.

절 제목 밑에 줄 긋기

예컨대 다음과 같이 하면 가운데 정렬하고 표제 영역에만 밑줄을 그을 수 있다.

```
\usepackage[normalem]{ulem}
\setsecheadstyle{\normalfont\sffamily\Large\centering\uline}
```

memoir 매뉴얼 [14]에 있는 다음 예제는 절 표제를 식자한 다음 그 길이에 상관없이 \textwidth 만큼의 선을 하나 그어준다. 이런 경우라면 \hrule도 괜찮을 것이다.

```
\newcommand\ruledsec[1]{%
  \Large\bfseries\raggedright #1 \rule{\textwidth}{.4pt}}
\setsecheadstyle{\ruledsec}
```

항번 부분만 여백 영역으로 끌어내리기

간단히 \hangsecnum을 선언하면 된다. 원래대로 되돌리려면 \defaultsecnum을 선언한다.

9.4 장 표제 아래 머리주

장 표제나 목차의 장 표제행 아래 짧은 개요를 두고 있는 경우가 있다. 이것도 주석의 일부로서 머리주 또는 두주(頭註)라 부르기도 한다. 주로 편집상의 주석의 역할을 하기 위해서 또는 장의 내용을 미리 알리거나 용어 설명 및 주의사항 등을 기록하는 데 쓰인다. 이것은 `\chapterprecis` 명령으로 구현할 수 있다. 이와 관련된 매크로를 요약하면 다음과 같다.

`\chapterprecis`	인자를 장 머리주로 식자하면서 목차에 추가한다.
`\chapterprecishere`	머리주를 식자만 한다.
`\chapterprecistoc`	머리주를 목차에만 추가한다.
`\prechapterprecis`	머리주를 식자하기 전에 놓일 텍스트 또는 코드이다.
`\postchapterprecis`	머리주를 식자한 후에 올 텍스트 또는 코드이다.
`\precistoctext`	목차에서 머리주가 식자될 때의 처리를 담고 있는 코드이다.
`\precistocfont`	목차에서 머리주가 식자될 때의 폰트이다.

이 매크로들은 머리주가 필요한 경우 아주 요긴하게 활용할 수 있다.

9.5 목차 안의 장절 표제

장절 표제를 식자하는 데 있어서 고려해야 할 점은 또 있다. 목차 안에서 장절 표제가 어떻게 식자되느냐 하는 것이 그 하나이고 다른 하나는 면주에서 장절 표제를 어떻게 나타낼 것이냐 하는 것이다.

목차를 식자하는 것은 라텍에 의하여 자동화되어 있지만 모든 설정이 디자이너가 의도하는 바를 모두 충족해주는 것은 아니다. 그러므로 목차의 편집 스타일을 결정하는 것은 민감하고 어려운 문제인데 여기에 대해서 memoir는 몇 가지 방법을 제시하고 있다. memoir 매뉴얼 [15, p. 129–142]에 목차 조판에 대한 상세한 안내가 있으므로 이 글에서는 이 문제를 더 다루지 않겠다.

9.6 새 목록류 만들기

memoir의 재미있는 기능 중의 하나가 새로운 목록류를 쉽게 만들 수 있다는 것이다. 이것은 memhangul-x와는 큰 관련이 없지만 이 클래스의 특징 중의 하나이므로 간단히 요약해두고 가기로 한다. 역시 더 자세한 것은 memoir 매뉴얼 [14]을 참고하기 바란다.

여기서는 이호재의 hozemanucs에 있는 `\query`라는 목록류 명령의 구현을 검토한다.

```
% query
\newcommand\queryfont{\slshape}
```

`\query`는 기울임체로 식자한다.

```
\newcommand{\query}[1][\@empty]{%
  \stepcounter{query}%
  \fbox{?}%
  \ifx #1\@empty
```

```
7       \addcontentsline{loq}{query}{\thequery)\thesection}%
8     \else
9       {\queryfont #1}%
10      \addcontentsline{loq}{query}{\thequery)\thesection~#1}%
11    \fi
12  }
```

\query 명령을 정의한다. 확장자가 .loq인 파일을 별도로 쓰도록 \addcontentsline 명령의 첫째 인자로 loq가 사용되었다. 세 번째 인자로 쓰인 \thequery를 위해 별도로 카운터를 정의할 필요는 없다. 대신 \newlistentry 명령이 직접 카운터를 만들어준다. 필요하다면 별도의 카운터를 만들어서 쓸 수도 있고 이미 정의된 카운터를 사용해도 된다.

```
13  \newcommand\loqname{List of Query}
14  \newlistof{listofqueries}{loq}{\loqname}
15  \newlistentry{query}{loq}{0}
```

이 부분이 새로운 'listof'를 정의하는 부분이다. \loqname은 query 목록을 만들 때 식자할 이름이다. 이것을 식자하는 이름은 \listofqueries로 되어 있으며 백슬래시 없이 \newlistof의 첫 번째 인자로 지정되어 있다. \newlistentry 명령은 \query 명령으로 엔트리를 만든다는 것인데 역시 백슬래시 없이 첫 번째 인자로 주어진다. 그리고 \query는 앞에서 이미 정의되어 있다. \newlistentry 명령은 한 개의 옵션 인자를 가질 수 있는데 그것은 여기서 정의되는 query 카운터가 종속되는 상위 카운터를 의미한다. 예컨대

```
\newlistentry[section]{query}{loq}{0}
```

이라고 정의하였다면 query 카운터는 \section이 바뀔 때마다 갱신될 것이다.

14번 줄에서 쓰인 \newlistof 명령은 \loqmark, \loqheadstart, \printloqtitle, \afterloqtitle이라는 네 개의 새로운 명령을 쓸 수 있게 해준다. 이들은 모두 query 목차를 만드는 데 사용된다. 이 명령들의 이름에 공통으로 쓰인 'loq'는 \newlistof의 두 번째 인자로 주어진 것이다. 그리고 loqdepth라는 카운터도 하나 생겨나는데 이것은 tocdepth처럼 식자하는 레벨을 의미한다. 기본값은 1이다.

```
16  \addtodef{\insertchapterspaces}{}%
17            {\addtocontents{loq}{\protect\addvspace{10pt}}}
```

이것은 장이 바뀔 때마다 query 목록에도 약간의 여분이 생기도록 설정하는 코드이다.

이제 memoir 클래스가 목차 조판을 위해 제공하는 '\cft...' 형태의 매크로들을 query에 대해서도 쓸 수 있게 되었다. 또한 \thequery를 재정의함으로써 query 카운터의 표현 형식을 바꿀 수 있다. 필요하다면 다음 명령을 이용하여 이 카운터를 특정 카운터에 종속시키거나 반대로 종속을 해제하는 것도 가능하다.

```
\counterwithin{query}{section}
\counterwithout{query}{section}
```

이 매크로의 별표붙은 버전은 query 카운터의 종속성은 바꾸면서 카운터의 표현 형식은 변하지 않도록 한다.

memoir 매뉴얼 [14]에는 \newlistof를 이용하여 subsubpara라는 새로운 하위수준 장절명령을 추가하는 예제가 있다. 아래에 그 코드를 인용해둔다.

```
\newcommand{\subsubpara}{\@startsection{subsubpara}
   {6}
   {\parindent}
   {3.25ex \@plus1ex \@minus.2ex}
   {-1em}
   {\normalfont\normalsize\itshape}
}
\newlistentry[subparagraph]{subsubpara}{toc}{5}
\cftsetindents{subsubpara}{14.0em}{7.0em}
\newcommand*{\subsubparamark}[1]{}
```

10 결론

oblivoir 매뉴얼은 실제 단행본을 디자인하기 위하여 필요한 요소를 갖추어야 한다. 그리고 이를 위해서는 상당히 많은 이론적 배경을 요구하는 것을 보았다. 우리는 그 가운데 특히 XeTeX-k*o*를 이용한 조판에 있어서 책의 기본 구성요소, 글꼴과 한글 타이포그래피, 판면설정에 관한 사항을 알아보았다.

이 사용설명서 작성의 시도를 발전시켜 실제적으로 도움이 되는 oblivoir 매뉴얼이 작성될 수 있기를 바란다. 그리고 무엇보다 중요한 것은 어떤 모양이나 기능을 어떻게 구현할 수 있느냐보다 왜 그 부분에서 그런 모양을 선택하는가에 대한 관점이 요구된다는 점을 확인한 것이다.

XeTeX과 k*o*.TeX의 놀라운 조판 기능을 이용하여 더욱 다양하고 쓸모있는 훌륭한 도서가 제작되기를 바라면서 글을 맺는다.

참고 문헌

1. 권영성, 《신고 헌법학개론》, 법문사, 1989.
2. 김강수, 한글 문장부호의 조판 관행에 대하여, The Asian Journal of TeX **1** (2007), no. 1, 17–30.
3. ———, 초간단 xoblivoir under XeTeX 사용법, 2008; 2011.
4. 김도현, 유니코드 k*o*.TeX에서 finemath 기능의 구현, The Asian Journal of TeX **1** (2007), no. 2, 135–151.
5. ———, XeTeX-k*o* 간단 매뉴얼, 2011. `http://ftp.ktug.or.kr/KTUG/texlive/texmf-dist/doc/xelatex/kotex-dev/xetexko/xetexko-doc.pdf`
6. 김창수, 출판인의 한글 서체 선택 이유와 서체 이미지에 관한 연구, Master's thesis, 경희대학교 언론정보대학원, 2004.
7. 안상수, 한글 타이포그래피의 가독성에 관한 연구, Master's thesis, 홍익대학교 대학원, 1980.
8. 유정미, 《잡지는 매거진이다》, 효형출판, 2002.
9. 이기황, Oblivoir를 이용한 문서 작성, The Asian Journal of TeX **1** (2007), no. 2, 123–134.
10. 이종운, 《도서편집총람 — 판면편집과 교정》, 범우사, 1991.

11. Umberto Eco, 김운찬 (옮김), 《논문 작성법 강의》. 열린책들, 1994.

12. Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley, Christine Detig, and Joachim Schrod, *The LaTeX Companion*, 2nd ed., Addison-Wesley, 2004.

13. Will Robertson and Khaled Hosny, *The fontspec package*, 2011. `CTAN:macros/latex/contrib/fontspec/fontspec.pdf`

14. Peter Wilson, *The Memoir Class for Configurable Typesetting: User Guide*, 8th ed., The Herries Press, 2010. `CTAN:macros/latex/contrib/memoir/memman.pdf`

15. Peter Wilson, 김강수 (옮김), 《memoir 클래스 사용자 설명서》, 6th ed., 2004; 2006. `http://doc.ktug.or.kr/memhangul/memucs-manual-all.pdf`