이 문서에 관하여

이 책은 공주대학교 문서작성워크숍 *2022*에서 필자가 담당한 "문서 스타일링에 관한 거의 모든 것"의 독본(*reading*) 자료로서 작성하였다.

필자가 작성한 pgreenbook의 안내문서 *pgreenbook-doc*의 내용을 근간으로 하여, 그 중 해당 패키지에 한정된 설명을 제외하고 일반적으로 요구되는 정보를 확장하였다.

이 문서에서는 장(chapter)이 있는 단행본 규모의 책을 중심으로 논의를 전개한다. 따라서 oblivoir에 `[chapter]` 옵션이 주어진 상태임을 알아두자.

사적 문서나 작은 보고서와 같이 section으로 시작하는 문서라면 chapter 관련 부분을 제외하면 유용할 정보일 것이다.

차 례

1

글자

문서는 '글자'로 이루어진다. 주로 한글로 된 문서를 작성하는 문제를 다룰 것이므로 대상으로 하는 글자는 한국어의 한글, 그리고 영어의 라틴 문자이다.

LaTeX에서 글자는 text와 math로 구분한다. 이 장에서는 text에 해당하는 사항만 살펴볼 것이다.

1 폰트: NFSS

LaTeX이 폰트를 지정하고 활용하는 방식을 NFSS (New Font Selection Scheme)이라고 하는데, 여기서 'new'라 함은 LaTeX2.09에 비하여 LaTeX 2_ε가 새롭다는 것이었으나 이로부터 이미 수십 년이 지난 지금 'new'는 조금 민망한 느낌이 있지만 이 구조를 통칭하는 표현이다.

텍스트 폰트는

- encoding: OT1, T1, LUC
- family: rm, sf, tt
- shape: up, it, sl, sc
- series: md, bf
- size

라는 범주로써 지정한다. 이 가운데 encoding에 관한 문제는 Legacy TeX에서 중요하였으나 유니코드 폰트를 주로 사용하게 된 현재는 일단 무시해도 좋다. (단 type 1이나 mf 글꼴을 사용하는 경우에는 매우 중요하다.)

family는 기본 3개: rm, sf, tt가 있다. rm은 roman, sf는 sans serif, tt는 typewriter를 의미하는데, 이 용어는 fontspec에 의하여 각각 mainfont, sansfont, monofont에 대응하는 것이다. 우리말로는 명조(바탕), 고딕(돋움)이 rm, sf에 해당하며 tt는 주로 소스코드 리스팅에 사용하는 것으로 다른 조판도구에서는 드문 family이다.

shape는 폰트의 '모양'을 가리킨다. upshape는 곧추선 모양, itshape는 이탤릭(흘림) 모양, slshape는 slanted(기울임) 모양, scshape는 small capital(작은 대문자) 모양이다. itshape와 slshape는 가끔 넘나들기도 하는데 그것은 해당하는 폰트 디자인이 없기 때문이다.

```
{\rmfamily Good} {\sffamily Good} {\ttfamily Type}.\\
{\rmfamily \upshape The more, the better.}\\
{\rmfamily \itshape The more, the better.}\\
{\rmfamily \slshape The more, the better.}
```

Good **Good** `Type`.

The more, the better.

The more, the better.

The more, the better.

series는 주로 무게(weight)에 관련된 것으로, md (Medium), bf (BoldFace)가 주로 쓰인다.

family, shape, series를 지정하는 선언형 매크로는 다음과 같다.

```
\rmfamily, \sffamily, \ttfamily,
\upshape, \itshape, \slshape, \scshape,
\mdseries, \bfseries
```

LaTeX에서는 편의를 위하여 이 선언형 매크로에 상당하는 인자형 명령을 정의해두고 있다.

lshort-ko p.30의 '강조' 부분을 읽고, 왜 문서의 본문에서 `\textit`을 쓰지 않고 `\emph`를 써야 하는지 설명하라.

```
\textrm{...}, \textsf{...}, \texttt{...}
\textup{...}, \textit{...}, \textsl{...}, \textsc{...}
\textmd{...}, \textbf{...}
```

size 명령은 잘 아는 대로 다음과 같다.

```
\miniscule, \tiny, \scriptsize, \footnotesize, \small
\normalsize,
\large, \Large, \LARGE, \huge, \Huge, \HUGE
```

이 가운데 `\miniscule`과 `\HUGE`는 memoir에서만 쓸 수 있다. size 명령으로 인쇄되는 실제 크기가 몇 포인트가 되는가는 전적으로 클래스와 옵션에 좌우된다. 예컨대 `\normalsize`의 경우 `[11pt]` 옵션을 주었을 때는 10.98pt, `[12pt]` 옵션을 주었을 때는 12.0pt이다. 이 명령들은 문서 전체의 옵션에 대한 상대적 크기임을 알아두자.

memoir 클래스에서 `\documentclass`에 부여할 수 있는 폰트 크기 옵션이 어떤 것이 있는지 조사해보자. 표준 LaTeX 클래스는 10pt, 11pt, 12pt 세 종류가 있다.

폰트 크기 매크로는 인자형 명령을 제공하지 않는다. 즉 `\textsmall` 같은 명령은 정의되어 있지 않다. 따라서 일부 단어의 글자 크기를 바꾸려면 다음과 같이 입력하여

```
{\tiny 바꾸려면}
```

영향을 미치고자 하는 단어를 범위지정하여야 한다. (중괄호로 둘러싸야 한다.)

절대 사이즈로 폰트 크기를 지정하는 방법이 없는 것은 아니지만 특히 행간격과 관련하여 주의해야 할 점이 많다는 사실을 알아두는 것이 좋다. 다음 보기의 두 번째 인자가 행간격(baselineskip)인데 실제 행간격이 효과를

발휘하는 것은 문단이 청산되는 때이다 (즉 문단 내의 한두 단어나 글자에서는 효과가 나타나지 않는다).

```
\fontsize{24.8pt}{26pt}\selectfont
```

2 폰트: fontspec

XeTeX, LuaTeX의 시대인 오늘날, 폰트 활용은 거의 전적으로 fontspec 패키지에 의존한다. 한국어 패키지인 XeTeX-ko와 LuaTeX-ko는 한글 및 한자 폰트를 지정할 수 있는 fontspec 스타일의 명령군을 제공한다.

- fontspec 매뉴얼
- XeTeX-ko 매뉴얼
- LuaTeX-ko 매뉴얼

시스템에 설치된 트루타입, 오픈타입 폰트를 사용하게 되면서 다른 고민이 생겨났는데 그것은 사용자마다 설치되어 가용한 폰트 세트가 다 다르다는 것이다. 그러므로 우선적으로 TeX Live로 배포되는 폰트로 이용 가능한 것이 무엇인지 조사해 보는 것이 중요한데 이것도 그 종류가 무척 많다. 예컨대 Libertine, STIX2, TeX Gyre 등의 폰트들은 이미 설치되어 있는 것이다.

한글 폰트는 다음 두 세트가 있다.

- 은 글꼴 트루타입 (unfonts-core, unfonts-extra)
- 백묵 트루타입 (baekmuk)

2.1 폰트 이름, 파일 이름

fontspec 명령으로 폰트를 지정할 때 두 가지 방법이 있다: 하나는 "폰트 이름으로 부르기"이고 다른 하나는 "파일 이름으로 부르기"이다. 다음은 폰트 이름으로 부른 것이고

```
\setmainfont{Noto Serif KR Light}
```

파일 이름으로 부르면 다음과 같이 된다.

```
\setmainfont{NotoSerifKR-Light.otf}
```

두 방법의 차이는 특히 확장자를 명시하는가 그렇지 않은가의 차이라고 생각하면 된다.

폰트 이름을 어떻게 확인하는가? 다음이 그 방법이다.

```
otfinfo -a C:\Windows\Fonts\NotoSerifKR-Light.otf
```

그러면 "Noto Serif KR Light"라는 이 폰트의 "폰트 이름"을 보여줄 것이다. otfinfo라는 유틸리티는 TeX Live로 설치된다. 대략적으로 말해서 Windows의 '글꼴 설정'에서 보여주는 폰트 이름이 이것과 일치하는 경우가 많지만 일부 차이도 있는데, 되도록 저 otfinfo 명령으로 확인되는 폰트 이름을 쓰도록 하는 것이 좋다.

폰트 이름으로 부르는 것과 파일 이름으로 부르는 것의 차이는 무엇인가?

(1) 대체로 "폰트 가족"의 자동 검출에는 폰트 이름으로 부르는 것이 낫다.

(2) 파일 이름으로 부르는 것이 "폰트 찾지 못함" 에러가 발생할 가능성이 낮다.

KTUG의 설치 가이드는 XeTeX, LuaTeX 어느 쪽이든, 그리고 폰트 이름이든 파일 이름이든 모두 부를 수 있도록 설정하는 방법을 상세히 설명하고 있다. KTUG 설치 가이드가 표준적인 설치 방법 안내에 비하여 더 복잡한 이유 중의 하나이다. 만약 KTUG 가이드를 충실히 따라 수행했다면 폰트 불러오기에서 문제가 발생할 일은 없을 것이다.

2.2 폰트 가족 설정

보편적으로 italic, bold, bolditalic 폰트를 지정하는 것으로 하나의 폰트 가족이 된다.[1]

```
\setmainfont{pala.ttf}[
  BoldFont={palab.ttf},
  ItalicFont={palai.ttf},
  BoldItalicFont={palabi.ttf}]
```

한글 폰트는 italic 폰트가 없다. 그런데 `\textit`을 식자해야 할 때는 거기에 해당하는 폰트를 제공해야 한다. 만약 이탤릭을 써야 할 곳에 UnGraphic.ttf를 식자하려 하는 경우라면

```
\setmainhangulfont{UnBatang.ttf}[
    ItalicFont={UnGraphic.ttf},
    BoldFont={UnBatangBold.ttf},
    BoldItalicFont={UnGraphicBold.ttf}
]
```

이렇게 할 수 있을 것이다.

본문 폰트를 이와 같이 설정하고 간단한 문단을 작성해보라.

fontspec은 폰트의 특성이나 속성에 따라서 매우 복잡한 설정을 가능하게 하고 있다. 이것은 오늘날의 OpenType이 가진 거의 대부분의 features를

[1] 예시한 Palatino Linotype 폰트는 자동으로 폰트 가족을 찾아준다.

LaTeX에서도 활용할 수 있다는 것을 의미한다. 여러 폰트 속성의 세부적 활용에 대한 것은 이 강좌의 목적이 아니고 이 자체가 하나의 토론 주제가 되므로 여기서는 단지 Scale 옵션에 대해서만 간략히 지적하고 간다.

```
[Scale=MatchUppercase]
```

이 옵션을 mainfont에 부여하면 "기존의 mainfont"의 Uppercase에 맞추어서 폰트를 확대/축소한다.

pgreenbook의 영문자 폰트는 다음처럼 설정하고 있다.

```
\setmainfont{STIX Two Text}
\setsansfont{Source Sans Pro}
    [Scale=MatchUppercase,BoldFont={* Semibold}]
```

여기서는 sans 폰트의 Scale 옵션으로 MatchUppercase를 주었기 때문에 기존의 sans 폰트(Latin Modern Sans)의 Uppercase에 맞추어서 스케일링해준다. 문장 중간에 fontfeature를 바꿀 수 있는데 그럴 경우에도 이 스케일링 옵션을 쓰는 것이 판면을 안정시키는 데 기여한다.

2.3 한글 폰트

표준적인 LaTeX의 설계는 상당히 보수적이다. 예컨대 본문 기본 글꼴은 명조체인 것을 당연하게 가정하고 있다. 복잡한 폰트 세트를 자신의 취향에 맞추어 설계하는 것은 알아서 할 일이지만, 이것저것 잘 모르겠으면, 다음 규칙을 추천한다.

(1) 한글 본문 폰트는 Noto Serif 또는 KoPub World Batang.

(2) 한글 돋움 폰트는 Noto Sans 또는 KoPub World Dotum.

이 두 폰트는 모두 다운로드하여 설치하여야 한다.

2.4 수식 내의 한글 폰트

수학식 내에서 한글을 `\text` 명령 없이 쓰려면 `\setmathhangulfont`를 절정해두어야 한다. unicode-math를 쓰지 않는 경우에도 그러하다.

2.5 한자 fallback 폰트

대부분의 한글 폰트는 2360자 (또는 11172자)의 현대 한글과 4888자 정도의 (1수준) 한자를 포함하고 있다. 어떤 폰트는 한자가 없다.

한자를 본격적으로 사용하는 문서에서 이것은 부족하다. 예컨대 jiwonlipsum의 한자 본문을 식자하는 데도 1수준 한자로는 부족하다. 그래서 한자가 많다면 더 많은 한자를 제공하는 폰트를 fallback 폰트로 설정하여 만약 기본 폰트에서 한자를 찾을 수 없다면 이 폰트로 식자하라고 설정해주어야 한다.

XeTeX-ko에서는 `\fallbackhanjafont`라는 폰트 패밀리를 설정하면 되고 LuaTeX-ko는 폰트에서 글자를 찾는 방식이 다르므로 mainhanjafont를 충분한 한자를 제공하는 폰트로 지정해두면 될 것이다.

2.6 부분적인 폰트 사용

XeTeX-ko를 기준으로 말하면, `\fontspec` 명령은 라틴 문자에만 적용된다. 한편 `\hangulfontspec` 명령은 한글에만 적용된다. 그러므로 이 두 switch 명령을 중복해서 사용해야 원하는 결과를 얻는 경우가 많다. 이 명령들을 사용할 때는 적용 범위를 잘 설정해야 한다.

```
문서 중간에
{\fontspec{texgyreadventor-bold.otf}%
 \hangulfontspec{UnShinmun.ttf}%
임시로 FONT를 바꿀 때}

문서 중간에 임시로 FONT를 바꿀 때
```

2.7 자간과 어간

오늘날 자간과 어간은 폰트의 속성으로 취급한다. fontspec의 LetterSpace, koTeX의 InterHangul 값을 조정할 수 있다. 어간은 fontspec의 WordSpace를 이용한다. (단 WordSpace는 전역적 효과를 가지므로 주의해서 설정하라.)

```
\addhangulfontfeature{InterHangul=3pt}%
\jiwon[3]
```

지 금 나 는 밤 중 에 한 강 을 아 홉 번 건 넜 다. 강 은 새 외 로 부 터 나 와 서 장 성 을 뚫 고 유 하 와 조 하·황 화·진 천 등 의 모 든 물 과 합 쳐 밀 운 성 밑 을 거 쳐 백 하 가 되 었 다. 나 는 어 제 배 로 백 하 를 건 넜 는 데, 이 것 은 하 류 였 다.

2.8 장평

장평은 오늘날 폰트의 Fake 속성으로 취급한다. 수능 시험지의 기본 폰트는 신명중명조, 자간 -5%, 장평 95%라고 한다. 신명중명조라는 폰트는 HWP의 HFT 폰트이므로 트루타입을 찾을 수 없다. (그렇다면 수능 시험지는 어떤 조판도구로 조판하고 있을까?) 예전 최정호 선생 디자인의 #중명조는 유료 폰트로 어디선가 구매가 가능하다고 한다.

```
\addhangulfontfeature{FakeStretch=.9}%
지금 나는 밤중에 한 강을 아홉 번 건넜다.\\
\addhangulfontfeature{FakeStretch=.8}%
지금 나는 밤중에 한 강을 아홉 번 건넜다.\\
\addhangulfontfeature{FakeStretch=1.5}%
지금 나는 밤중에 한 강을 아홉 번 건넜다.
```

지금 나는 밤중에 한 강을 아홉 번 건넜다.
지금 나는 밤중에 한 강을 아홉 번 건넜다.
지금 나는 밤중에 한 강을 아홉 번 건넜다.

3 oblivoir의 폰트 설정 명령

oblivoir는 다음 폰트 설정 명령을 더 제공한다.

```
\setkomainfont[...](<main>)(<bold>)(<italic>)[option]%
    [...](<main>)(<bold>)(<italic>)[option]
\setkosansfont
\setkomonofont
```

이 명령의 첫 옵션 인자는 공통 부분을 줄여쓰려는 목적으로 쓴다. 뒤에 붙는 옵션 인자는 fontspec options. 여기에는 중괄호가 하나도 없다는 데 주의하여야 한다.

```
\setkomainfont[Noto Serif KR]( Light)( Bold)( Medium)[Script=Hangul]
```

이것은 다음처럼 한 것과 동일하다.

```
\setmainhangulfont{Noto Serif KR Light}[%
  BoldFont={Noto Serif KR Bold},
  ItalicFont={Noto Serif KR Medium},
  Script=Hangul
]
```

라틴 문자 글꼴에 대하여 \setobmainfont를 써서 같은 신택스를 사용할 수 있다.

4 pgreenbook의 폰트 설정

pgreenbook은 다음처럼 폰트를 설정하고 있다.

```
\setmainfont{STIX Two Text}
\setsansfont{Source Sans Pro}
    [Scale=MatchUppercase,BoldFont={* Semibold}]
\setkomainfont[KoPubWorldBatang ](Light)(Medium)
\setkosansfont[KoPubWorldDotum ](Light)(Medium)
\ifXeTeX
\newfontfamily\fallbackhanjafont{Noto Serif KR}
    [Scale=.92]
\xetexkofontregime[puncts=prevfont, colons=prevfont,
    cjksymbols=hangul]{latin}
\fi

\usepackage{unicode-math}
\setmathfont{latinmodern-math.otf}
\setmathhangulfont{KoPubWorldBatang Light}
```

라틴 문자 메인	STIX Two Text
라틴 문자 산세리프	Source Sans Pro
한글/한자 메인	KoPub World Batang
한글/한자 산세리프	KoPub World Dotum
수학 기본	Latin Modern Math
수학 한글	KoPub World Batang

이 스타일에서는 `\xetexkofontregime`으로 문장부호의 식자 폰트를 지정했는데, 이 명령은 XeTeX-ko에서만 쓸 수 있다. 대체로 이 설정이 없어도 큰 위화감 없이 잘 나온다. 자세한 설명은 XeTeX-ko 매뉴얼에 나오므로 이를 숙지한 후에 디자인 의도를 구현하는 명령을 작성하라.

2

페이지

1 판형과 페이지 파라미터

oblivoir의 부속 패키지인 fapapersize는 간단하게 판형을 설정하도록 도와주는 것이다. 사용법은 oblivoir manual에 상세하다.

• oblivoir 매뉴얼

stock이나 trimmarks는 옵셋 인쇄를 위한 pdf를 준비할 때 필요한 것이다. 일반적인 개인용 문서, 보고서, 온라인 문서 등에는 불필요하다.

다음 pgreenbook의 예는 본격적인 인쇄물을 제작하기 위한 것이므로 stock size를 별도로 지정하고 trimmarks를 그렸다.

memoir가 제공하는 trimmarks의 종류를 조사해보자. oblivoir는 여기에 `\trimKmarks`가 추가되어 있다.

```
\usepackage[stock]{fapapersize}
\usefastocksize{210mm,297mm}
\setheadfoot{\headheight}{12mm}
\usefapapersize{174mm,251mm,27mm,*,35.5mm,*}
\quarkmarks
```

한편, 이 책은 다음과 같이 설정한다.

```
\usepackage{fapapersize}
\usefapapersize{188mm,257mm,25mm,53mm,25mm,32mm}
```

A시리즈 용지 (A4, A5, A6, …)와 B시리즈 용지 (B4, B5, B6, …)는 크기에 익숙할 것이다.

판형	작업 사이즈	재단 사이즈
국배판(A4) 210*297	216*303	210*297
국배판(A4) 210*280	216*286	210*280
국배판(A4) 190*260	196*266	190*260
신국판 152*225	158*231	152*225
국판(A5) 140*205	146*211	140*205
국판(A5) 148*210	154*216	148*210
B4 250*353	256*359	250*353
B5 182*257	188*263	182*257
46배판 188*257	194*263	188*257
크라운판 176*248	182*254	176*248

'크라운판'이라 불리는 사이즈는 보통 175×248mm (18절)로 재단한다. pgreenbook의 사이즈는 크라운판에 가까운 변형판본이다. 인쇄용지를 A4로 보고 재단선을 긋고 있다. 한편 이 책은 국배판 사이즈의 한 유형이다.

이 디자인에 의하면 글줄 길이(`\textwidth`)가 110mm 가량이 된다. 그 대신 오른쪽 마진을 충분히 넓게 잡아서 marginpar를 활용하려 하였다. 이런 디자인을 '변2단 조판'이라 부르는 경우도 있으나 사실상 2단은 아니다.

이 페이지의 길이값들을 계산하여 찍어보자. (표 2.1)

페이지를 구성하는 파라미터는 memoir 매뉴얼 p. 11에 나오는 표 2.1을 잘 상고하기 바란다.

• memoir 매뉴얼

표 2.1: 이 페이지의 길잇값

\textwidth	109.65604mm
\spinemargin	25.00012mm
\foremargin	53.00027mm
\marginparwidth	48.0798mm

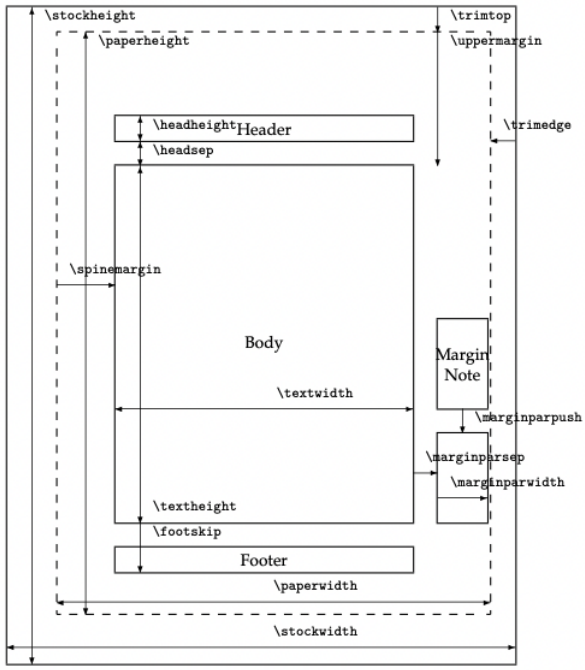


그림 2.1: memoir 홀수쪽 파라미터

이 파라미터들은 모두 memoir에 설명이 자세히 나와 있다. 만약 마진 문단을 디자인할 생각이라면 좀더 세심한 주의가 필요하다. `\marginparwidth` 및 `\sideparwidth` 등이 항상 `\spinemargin` 또는 `\foremargin`과 일치하지 않기 때문이다.[1]

memoir는 이 파라미터들을 조절하는 별도의 명령을 제공한다. 즉 파라미터 변수를 단순히 `\setlength`나 `\renewcommand`하면 뜻대로 되지 않을 수도 있다. 그리고 fapapersize는 이 가운데 가장 중요한 parameter들을 간단히 설정하게 도와주는 것이다.

[1] 최근 memoir에서 많이 개선되었다고 한다.

2 본문 줄 간격

```
\SetHangulspace{1.4}{1.12}
\RequirePackage[mathleading=1.2]{ob-mathleading}
```

`\SetHangulspace`는 oblivoir 명령이다. 두 개의 인자를 취하며 앞의 것이 본문 행송, 뒤의 것이 '좁은 행송'인데 이것은 특히 footnote와 floats 안의 간격을 제어한다. 자세한 사례는 oblivoir-test 문서를 참고하라.

행송은 `\linespread`값으로서 달리 `\baselinestretch`라고 부르는 값이다. oblivoir 자체 기본값은 1.333인데, 이 책에서는 1.4로 하고 있다. 이것이 무엇을 의미하는지 계산을 조금 해보자.

본문 10pt 문서에서 TEX은 12pt를 행간격 기본값으로 설정한다. 이것은 원래 폰트 속성으로서 예컨대 `\fontsize`명령의 두 번째 인자와 같다. 즉,

```
\fontsize{12pt}{14pt}
```

이 명령은 글자를 12포인트로, 윗줄과 아랫줄 baseline의 간격을 14포인트로 설정한다는 의미이다.

본문 10포인트의 경우, 즉 행 사이 12포인트를 고려할 때, 1.33이라는 값은 $12 \times 1.333 = 15.996$을 의미하게 되는데, 이것은 HWP의 기본 행간인 160%에 매우 근사한 값이다. 즉 oblivoir는 행간이 HWP와 매우 비슷하게 보이도록 디자인되어 있는 것이다. 이 책의 1.4는 16.8포인트로서, 168%에 상당하는 값이다. 이 정도 간격이 한글 책 독자들에게 잘 받아들여진다고 한다.

소위 배행간이라 불리는 200%는 얼마나 값을 주어야 할까? 1.67 정도가 거의 정확하게 200% 행간격을 설정하게 한다.

여기서 설정하는 행간격은 문서 전체에 걸치는 값이다. 만약 특정 문단에만 행간을 달리 적용하고자 한다면(자동으로 처리해주는 footnote와 floats를 제외하고) memoir의 *Spacing*이나 oblivoir의 *spacing* (이 둘은 본질적으로 같다)을 이용하면 된다.

ob-mathleading 패키지는 amsmath-류 여러 줄 수식의 행 사이 간격을 제어하는 패키지이다. 본문보다는 좁은 행간을 지정해야 보기 좋은 수식이 나온다.

• ob-mathleading 설명서

```
texdoc ob-mathleading
```

명령으로 설명서를 읽어보아라.

3 색상

문서에서 색상을 사용하기 위하여 필요한 패키지는 xcolor이다. 이밖에 이름으로 정의된 색상을 사용하기 쉽게 하는 패키지로

- xcolor 색상 이름 라이브러리 (x11names, svgnames, dvipsnames)
- ninecolors tabularray에서 쓰는 색상
- latexcolors

등이 있다. xcolor 색상 이름 라이브러리는 매뉴얼을 볼 것. •xcolor 매뉴얼

한편, pgreenbook에서는 새로운 색상을 정의해서 사용하는 방법을 보였는데 그것은 다음과 같다.

```
%% 색깔설정
\@ifpackageloaded{xcolor}{}{
    \RequirePackage[dvipsnames,svgnames,x11names]
                   {xcolor}
}
% Pantone Fashion Color: Spring 2014
\definecolor{PlacidBlue}{cmyk}{.47,.17,.02,.0}
\definecolor{VioletTulip}{cmyk}{.44,.39,.0,.0}
\definecolor{Hemlock}{cmyk}{.39,.04,.35,.0}
\definecolor{Paloma}{cmyk}{.35,.24,.27,.0}
\definecolor{Sand}{cmyk}{.20,.27,.48,.0}
\definecolor{Freesia}{cmyk}{.0,.14,.100,.0}
\definecolor{Cayenne}{cmyk}{.06,.74,.56,.0}
\definecolor{CelosiaOrange}{cmyk}{.0,.63,.80,.0}
\definecolor{RadiantOrchid}{cmyk}{.32,.65,.0,.0}
\definecolor{DazzlingBlue}{cmyk}{.92,.57,.0,.0}
\definecolor{PurpleHaze}{cmyk}{.56,.51,.15,.0}
\definecolor{Comfrey}{cmyk}{.74,.28,.63,.10}
\definecolor{MagentaPurple}{cmyk}{.51,.94,.24,.24}

\colorlet{MainColorOne}{Comfrey}
\colorlet{MainColorTwo}{PurpleHaze}
```

단행본 디자인에서 색상은 매우 중요하다. 인쇄물을 목적으로 하는 때와 온라인 출판을 의도하는 때에 사용할 수 있는 색공간이 다르기 때문이다. 인쇄물에 (변환되지 않은) RGB 삽도를 쓰면 색공간이 어그러지기 때문에 특별한 주의가 필요하다.

여기에서는 모두 cmyk 색공간으로 사용할 색을 지정하고 있다. 화면에서 보던 익숙한 색과 색감에 차이를 보이는 이유는 색공간이 cmyk이기 때문이다.

PlacidBlue
VioletTulip
Hemlock
Paloma
Sand
Freesia
Cayenne
CelosiaOrange

RadiantOrchid
DazzlingBlue
PurpleHaze
Comfrey
MagentaPurple
MainColorOne
MainColorTwo

pgreenbook이 '초록책'이라 불리는 이유는 주판면에 사용한 색상 `MainColorOne` 때문일 것이다.

만약 2도 인쇄 또는 별색(spotcolor)을 사용하는 책을 디자인한다면 거기에 맞춘 특별한 설계와 주의가 필요하나, 이 글에서 다룰 수는 없다. 이 책이 4도 인쇄를 위하여 디자인되었음을 지적하는 것으로 그친다.

4 memoir와 tikz

```
\RequirePackage{tikz}
\RequirePackage{tikzpagenodes}
%\RequirePackage{memtikzpagenodes}
%\RequirePackage{oblivoir-misc}
```

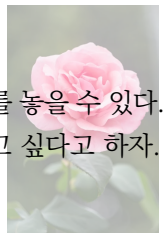
TikZ를 페이지 디자인에 활용하려 할 때, tikzpagenodes 패키지는 매우 요긴하다. 그런데 이 패키지를 그대로 쓸 때는 이것이 memoir의 페이지 요소와 일부 어긋날 수 있다는 것을 알아야 한다.

Kriss가 기여한 바, memoir+tikzpagenodes 호환을 위한 삽질기라는 글에 이 문제를 다루고 있고, memtikzpagenodes.sty를 제시하고 있다.

한편 tikzpagenodes를 쓰지 않고 단지 TikZ의 `current page` 노드만을 쓴다 해도 역시 비슷한 일이 일어난다. 이 문제는 oblivoir-misc 패키지를 로드하는 것으로 해결된다.

4.1 페이지 요소 디자인과 TikZ

TikZ를 이용하여 페이지 내의 어떤 위치에 원하는 오브젝트를 놓을 수 있다. 예컨대 이 페이지의 typeblock 하단 오른쪽에 꽃을 하나 두고 싶다고 하자.



만약 이후 모든 페이지, 또는 `\chapter` 명령이 주어지는 모든 페이지에 꽃 그림을 두려 한다면 어떻게 해야 할지 고민해 보자.

```
\begin{tikzpicture}[remember picture,overlay]
\tznode[opacity=.4,anchor=south east]
([xshift=-\foremargin,yshift=\lowermargin]
   current page.south east){\includegraphics[width=2cm]{rose}}
\end{tikzpicture}
```

이와 같이 그림, 선, 도형을 페이지 디자인 요소로 사용하려 할 때 TikZ는 거의 필수적이다. TikZ 및 이의 활용성 패키지인 tzplot에 숙달해두는 것은 비단 그림을 그리기 위해서만이 아니고 이러한 페이지 요소를 구현하기 위해서도 요구되는 것이다.

tzplot에 대해서는 오늘 강좌에서 공부할 것이다. 지금 보이는 이 화살표는 어떻게 그렸을지 소스를 보고 고민해보자.

3

장(chapter)과 페이지의 스타일

1 chapter style

장 스타일의 설계와 구현은 그림 3.1에 표현된 파라미터 매크로들에 적절한 값을 주거나 renewcommand하는 방식으로 이루어진다.

두 개의 사례를 보자. pgreenbook의 *KNUchapter*라는 장스타일은 다음과 같이 정의된 것이다. 그리하여 그림 3.2와 같은 모양을 얻었다.

```
\makechapterstyle{KNUchapter}{%
    \setlength{\afterchapskip}{40pt}
    \renewcommand*{\chapterheadstart}
        {\vspace*{-3\onelineskip}}
    \renewcommand*{\afterchapternum}
        {\par\nobreak\vskip 25pt}
    \renewcommand*{\chapnamefont}
        {\normalfont\LARGE\flushright}
    \renewcommand*{\chapnumfont}
        {\normalfont\HUGE\color{MainColorOne}}
    \renewcommand*{\chaptitlefont}
        {\color{MainColorOne}\sffamily\HUGE\bfseries\flushright}
    \renewcommand*{\prechapternum}
        {\chapnamefont\MakeUppercase{Chapter}}
    \renewcommand*{\postchapternum}{%
        \makebox[0pt][l]{%
        \hspace{1em}{\color{MainColorOne!50}%
        \rule{\midchapskip+\spinemargin}{\beforechapskip}}
    }}
    \renewcommand*{\chapternamenum}{}
    \setlength{\beforechapskip}{18mm}
    %\numberheight
    \setlength{\midchapskip}{\paperwidth}
    %\barlength
    \addtolength{\midchapskip}{-\textwidth}
    \addtolength{\midchapskip}{-\spinemargin}
    \renewcommand*{\printchapternum}{%
        \hspace{1em}\resizebox{!}{\beforechapskip}%
        {\chapnumfont \thechapter}}
    \makeoddfoot{plain}{}{}
        {\normalfont\normalsize\sffamily\thepage}
}
```

이 그림에 나오는 각 파라미터들을 어떤 식으로 재정의하여 이 책의 장 스타일을 만들었는지 잘 살펴보는 것이 공부가 될 것이다. 필요하다면 tikz를 활용하여 여러 도형이나 선을 긋거나 그림 등을 활용하여 꾸밀 수 있다. 어느 위치의 매크로를 재정의하는 것이 가장 좋을지는 그때그때 상황에 따라 다르므로 약간의 아이디어가 필요한 경우도 있다.

이 책의 chapter style은 이름이 `demo`이다. 다음과 같이 정의한 것이다.

```
\makechapterstyle{demo}{%
```

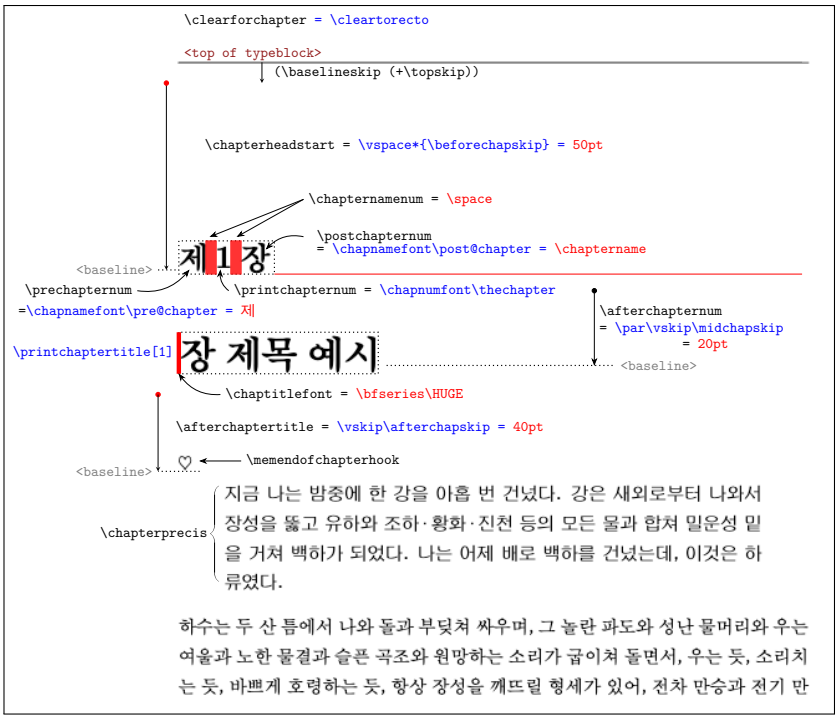


그림 3.1: oblivoir의 장 표제 파라미터

이름을 'demo'로 하고 정의를 시작한다. 마지막의 중괄호가 열린 것은 정의가 끝나는 위치에서 닫혀야 한다.

```
\renewcommand{\chapterheadstart}{%
    \vspace*{\beforechapskip}
    \begin{tikzpicture}[remember picture,overlay]
    \tznode<0,3mm>(current page.north){%
        \includegraphics
            [width=\dimexpr\paperwidth+6mm\relax,
             height=.3\paperheight]{rain}
    }[b]
    \end{tikzpicture}
}
```

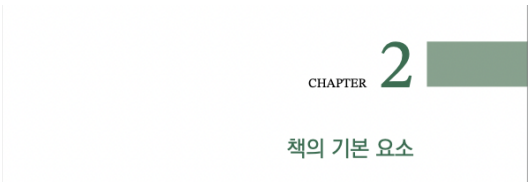


그림 3.2: pgreenbook의 KNUchapter

\chapterheadstart 매크로를 맨처음 재정의하였다. 그 핵심은 rain.jpg 라는 이름의 그림을 current page.north를 기준으로 배치하는 것이다.

```
\renewcommand{\prechapternum}{}
\renewcommand{\postchapternum}{}
\renewcommand{\chapternamenum}{}
\renewcommand{\printchapternonum}
    {\par\nobreak\vspace{\dimexpr\midchapskip+40pt\relax}}
```

chapternumber 관련 매크로 세 개를 아무 것도 찍히지 않게 만든 다음, \printchapternonum을 정의했는데 이 매크로가 하는 일은 '차례'와 같이 \chapter*가 실행되는 경우에 필요한 동작을 하게 하기 위한 것이다.

```
\renewcommand{\printchapternum}{%
    \vspace*{-50pt}
    \begin{adjustwidth}{0pt}{-\dimexpr\foremargin-3em\relax}
    \raggedleft\color{kmcyan!30!white}
      \fontspec{TeX Gyre Schola}
      \bfseries\itshape\fontsize{68pt}{70pt}\selectfont
      \thechapter
    \end{adjustwidth}
}
```

장 번호를 위치를 잡아서 식자하는 부분이다. 이 구현은 다른 방법, 예를 들자면 resizebox 같은 것을 이용할 수도 있다. 여기서는 단순하게 폰트 크기를 키웠다.

```
\renewcommand\chaptitlefont{\sffamily\bfseries\Huge}
\renewcommand\printchaptertitle[1]{%
    \begin{adjustwidth}{0pt}
        {-\dimexpr\foremargin-3em\relax}
    \raggedleft\chaptitlefont ##1
    \end{adjustwidth}
}
```

장 제목을 식자하는 부분인데, \printchaptertitle은 항상 한 개의 인자를 취하고 그것이 ##1로 지시되고 있다. 이 코드의 내용은 식자될 문단 폭을 넓힌 다음 오른쪽 끝에 장 제목을 가져다둔 것이다.

```
\renewcommand\memendofchapterhook{%
    \if@mainmatter
    \begin{adjustwidth}{0pt}{-\dimexpr\foremargin-3em\relax}
    \item
    \raggedleft
    \chaptertoc
    \end{adjustwidth}
    \cleartoverso
    \fi
}
```

```
}
```

\memendofchapterhook에 \cleartorecto가 있기 때문에 항상 chapter 표제면 다음 짝수면에서 본문이 시작한다. 이것을 \cleartorecto로 하면 장표제면 뒤에 백면을 두고 홀수면에서 시작하게 된다. 다만 장 표제면 자체가 항상 recto에서 시작하게 되는 기본값을 바꾸지 않았다.

chapter 표제면을 그냥 두기 심심해서 \chaptertoc를 식자하게 하였다. 이에 대해서는 따로 언급한다.

마지막의 닫는 중괄호로 \makechapterstyle 명령이 종료되었다.

한편 한글 서적에서는 이 책과 같이 장 표제면이라 하여 장 표제를 하나의 독립 페이지로 꾸미는 일도 많다. 이 경우 가장 간단한 방법은 \memendofchapterhook에 \clearpage 또는 \cleartorecto를 넣어주는 것이다. 대체로 단행본이 아니라면 이렇게 장 표제면을 별면으로 만들 필요는 없다.

chapter가 좌우 어디에서든 시작할 수 있게 하는 것이 바람직한가? 온라인 문서라면 그렇게 해야 할 필요도 있을 것이다. memoir에서 어떻게 하면 되는지 조사해 보자.

1.1 chaptertoc

여기에서 \chaptertoc 명령으로 장 내부의 절 제목과 페이지를 모아서 작은 toc를 작성하였다. 이것은 obchaptertoc라는 oblivoir 부수 패키지에 의한 것이다.

```
%%% chaptertoc
\RequirePackage{obchaptertoc}
\chaptertocmaxlevel{section}
\ChapterTOCafterskipfalse
\renewcommand\chaptertocfont{\small}
\ChapterTOCFormat{%
    \cftsetindents{section}{0pt}{1.5em}
    \renewcommand\cftsectionfont{\hfill}
    \renewcommand\cftsectionleader{,\:}
    \renewcommand\cftsectionafterpnum{\cftparfillskip}
    \renewcommand\cftsectionpagefont{\itshape}
}
```

패키지 문서를 읽을 수 있다. 또한 \cft... 명령에 대해서는 이 책의 4장에서 상세하게 논의하고 있다.

이 목적을 위한 다른 패키지도 몇 있다. 예를 들면 titletoc, minitoc, etoc 등. oblivoir가 아니라면 이런 종류의 패키지를 검토해보아야 할 것이다.

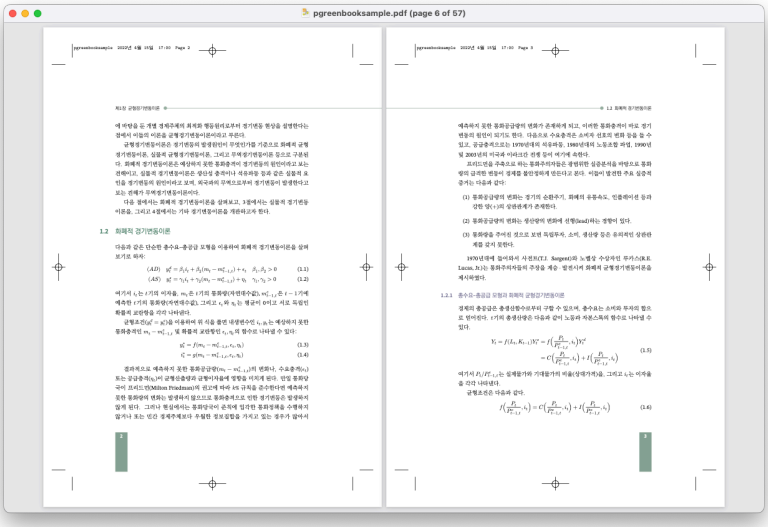


그림 3.3: KNUworkshop 페이지 스타일

2 페이지 스타일

pgreenbook은 두 종류의 페이지 스타일을 정의한다. 각각 KNUworkshop과 PageStylePrinciple이라는 이름으로 정의되었다. 시각적인 모양은 각각 그림 3.3과 그림 3.4에 나타나 있다.

2.1 페이지 스타일 기초

pagestyle을 정의하는 것은 어렵지 않다. 먼저 페이지스타일의 이름을 짓는다.

```
\makepagestyle{<pagestyle>}
```

하나의 페이지의 상단(head)과 하단(foot)에 각각 left, center, right로 배열되는 세 개의 '영역'이 있다고 생각하자. 그러면 모두 여섯 개의 '영역'에 무언가를 넣을 수 있다. 양면 조판의 경우 좌우 페이지가 달라지므로 12종류의 '뭔가 넣을 것'을 생각할 수 있고, 이것을 다음 매크로로 정의해주는 것이다.

```
\makeoddhead {<pagestyle>}{<left>}{<center>}{<right>}
\makeoddfoot {<pagestyle>}{<left>}{<center>}{<right>}
\makeevenhead{<pagestyle>}{<left>}{<center>}{<right>}
\makeevenfoot{<pagestyle>}{<left>}{<center>}{<right>}
```

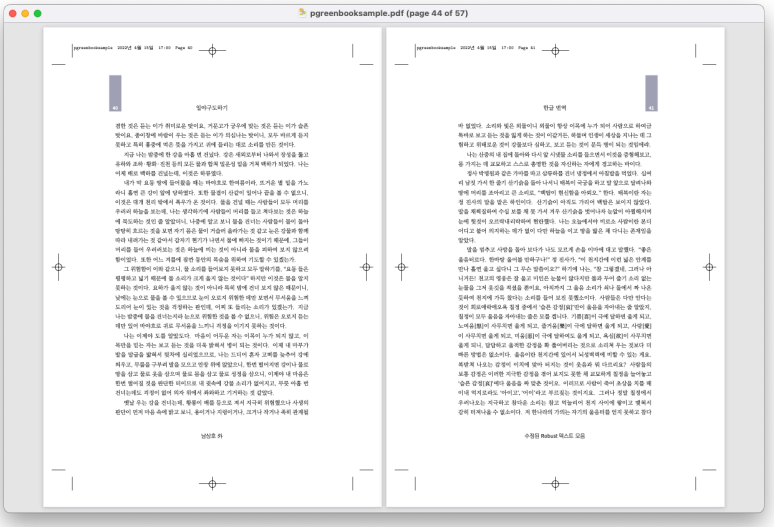


그림 3.4: `PageStylePrinciple` 페이지 스타일

'뭔가 넣을 것'은 뭐라도 좋다. 텍스트, 박스, tikz 드로잉 등이 주로 오는 것들이다.

추가로 고려할 것은 `\makerunningwidth`이다. 페이지의 `\textwidth`와 다른 길이를 가지도록 할 수 있어서 판면보다 넓은 헤딩을 작성할 수 있게 한다.

페이지 스타일 정의 부분의 소스 코드는 인용하지 않겠다, 너무 길어서. 스타일 파일의 해당 부분을 잘 공부해 보면 얻는 것이 있을 것이다.

2.2 running heading의 장절표제 텍스트

LaTeX에는 `\leftmark`와 `\rightmark`라는 중요한 매크로가 있다. 원래 아무 것도 들어 있지 않은데, `\markboth`와 `\markright`라는 명령에 의해 그 내용이 채워진다.

`\markboth` 두 개의 인자를 취하여, 앞의 것을 `\leftmark`에, 뒤의 것을 `\rightmark`에 할당한다.

`\markright` 한 개의 인자를 취하여 그것을 `\rightmark`에 할당한다.

중요한 것, ~~`\markleft`~~라는 명령은 없다는 사실. `\markleft`를 하고 싶다면 먼저 `\markboth`를 실행하고 `\markright`로 빈 인자(또는 다른 것)를 차례로 주면 된다.

이 두 명령은 문서 중에 사용하는 것이 불가능하지는 않지만, 표준 LaTeX에서는 \chaptermark나 \sectionmark 같은 다른 명령에 의해 불리도록 설계되어 있다. 즉, \chapter 명령은 그 자체가 \chaptermark를 부르게 되는데 그 내용은 \markboth를 실행하는 것이다. 한편 \section 명령은 그 자체가 \sectionmark를 부르게 되는데 그 내용은 \markright를 실행하는 것이다. 그 결과, \chapter와 \section이 차례로 실행되고 나면 chapter 헤딩 타이틀은 \leftmark 매크로에, section 헤딩 타이틀은 \rightmark에 들어가 있게 된다. 굳이 "헤딩 타이틀"이라고 한 이유는 장 표제와 헤딩 타이틀이 동일하지 않을 수도 있기 때문이다. 예컨대 \chapter[헤딩 타이틀]{장 표제}와 같은 명령을 부여하면 장 표제로 찍히는 것은 "장 표제"라는 문자열이지만 헤딩 타이틀은 "헤딩 타이틀"이라는 문자열이 된다. 이 때 \leftmark에 들어 있는 내용은 "헤딩 타이틀"이다.

memoir/oblivoir에는 굉장히 중요한 특징이 한 가지 있다. 즉 pagestyle마다 running heading을 다르게 찍히도록 할 수 있는 것이다. 그것은 \makepsmarks라는 것인데, 기본 명령꼴은 다음과 같다.

```
\makepsmarks{<pagestyle>}{<running-headings>}
```

그리고 이 명령의 두번째 인자 자리에서 사용할 \createmark라는 명령이 있다. memoir 매뉴얼에 이에 대한 자세한 설명을 꼭 한 번 읽어보기 바란다. 여기서는 이 책에서 사용하고 있는 범위에서 다음과 같은 사용법만을 소개하고 지나간다.

```
%% psmarks
\makepsmarks{KNUworkshop}{%
    \createmark{chapter}{left}{shownumber}
        {\pre@chapter}{\post@chapter\enskip}
    \createmark{section}{right}{shownumber}{}{\enskip}
}
\makepsmarks{PageStylePrinciple}{%
    \createmark{chapter}{left}{nonumber}{}{}
    \createmark{section}{right}{nonumber}{}{}
}
```

\createmark는 무려 다섯개의 인자를 취한다. 순서대로 (1) 문서 섹셔닝 레벨 (2) mark 위치 (3) 번호 형식 (4) 번호 앞에 오는 것 (5) 번호 뒤에 오는 것.

(1) 문서 섹셔닝 레벨. chapter, section, subsection 등

(2) mark 위치. left, right, both 셋 중의 하나.

(3) 번호 형식. `shownumber`, `nonumber`, `notitle` 셋 중의 하나. 이 가운데 `notitle`은 "`2.3`절 절제목"과 같은 러닝 헤딩 중에서 "`2.3`절"에 해당하는 부분만 표시한다는 뜻이다.

(4) 번호의 앞부분. 예를 들면 `chapter 1`에서 "chapter"에 해당하는 것이다.

(5) 번호의 뒷부분. 예를 들면 `chapter 1`과 `chapter title` 사이에 무엇을 둘 것인가, 즉 장번호의 숫자를 찍은 뒤에 둘 것이다.

`nonumber`의 경우에 4, 5번 인자는 단순히 빈 인자 `{}`로 둔다. 단 한 줄만 보자.

```
\createmark{chapter}{left}{shownumber}
    {\pre@chapter}{\post@chapter\enskip}
```

이것은 `KNUworkshop` 페이지 스타일에서, `\chapter` 명령이 불리면

```
\leftmark =
    \pre@chapter \value{chapter}\post@chapter
    \enskip <titletext>
```

이 되도록 하라는 뜻이 된다. 지금 이 위치에서

3 장 장(CHAPTER)과 페이지의 스타일

이것이 그 결과이고 실제로 짝수쪽 헤딩에 이렇게 나타나고 있을 것이다.

2.3 demo 페이지 스타일

이로부터 배운 내용을 토대로 이 문서의 페이지 스타일을 제작해보자.

```
\makepagestyle{demo}
\makeevenfoot{demo}
    {\begin{tikzpicture}[remember picture,overlay]
     \tzdot*[kmcyan,fill=kmcyan](1pt,0)(3pt)
     \tzline[kmcyan]
       (1pt,0)
       (\dimexpr\textwidth+\marginparsep+\foremargin\relax,0)
     \tznode(0,0){\sffamily\footnotesize\thepage}[l]
     \tznode(5pt,0)
       {\colorbox{white}{\sffamily\footnotesize\leftmark}}
       [r]
     \end{tikzpicture}
    }{}{}
```

kmcyan은 단순히 cyan의 이름만 바꾼 색상이다. 다른 원하는 색상으로 해도 좋다. 아주 간략히 이 코드를 정리하면 페이지 하단에 파란색 줄을 긋고

\leftmark와 \thepage를 적당한 위치에 놓은 것이다. 줄의 길이가 좀 길고 그래서 길이 계산이 복잡해보일 뿐이다.

```
\makeoddfoot{demo}
    {\begin{tikzpicture}[remember picture,overlay]
     \tzdot*[kmcyan,fill=kmcyan](\textwidth,0)(3pt)
     \tzline[kmcyan]
       (-\marginparsep-\spinemargin,0)(\textwidth,0)
     \tznode<2pt,0>(\textwidth,0)
       {\sffamily\footnotesize\thepage}[r]
     \tznode(\dimexpr\textwidth-5pt\relax,0)
       {\colorbox{white}{\sffamily\footnotesize\rightmark}}
       [l]
     \end{tikzpicture}
    }{}{}
```

이 역시 줄을 긋고 \rightmark와 페이지 번호를 찍은 것이 전부다.

```
\makepsmarks{demo}{%
        \createmark{chapter}{left}{shownumber}{\bfseries장}{\,\;}
        \createmark{section}{right}{nonumber}{}{}
}
```

\leftmark와 \rightmark를 생성하는 방법을 명시하였다.

2.4 몇 가지 보충

- hangul 페이지 스타일은 oblivoir에서 쓸 수 있는 한국식 페이지 스타일이다.
- \copypagestyle은 이미 있는 스타일을 복사하여 수정할 수 있게 한다.
- \aliaspagestyle은 특별한 목적의 페이지 스타일을 이미 있는 스타일과 같게 만들 수 있다.

중요한 것은 chapter라는 이름의 페이지 스타일인데, \chapter 명령이 불리는 단 한 페이지는 이 스타일이 된다. 기본값이 plain이므로 반드시 원하는 모양으로 alias해주어야 한다. 이 책에서는

```
\aliaspagestyle{chapter}{empty}
```

과 같이 하여 \chapter가 쓰이는 페이지의 모양을 empty로 하였다.

3 section 표제 스타일

section보다 낮은 수준의 문서 구분 명령의 표제 스타일은 `\setsecheadstyle`이나 `\setsechook`, `\secnumformat` 등을 사용하여 장식할 수 있다.

이 책의 section 스타일은 보기를 들기 위해 좀 복잡하게 되어 있는데, 다음과 같이 작성한 것이다.

```
%%% section
\newlength{\tmplen}
\setsechook{%
    \setsecnumformat{%
        \settowidth{\tmplen}{\thesection}
        \addtolength{\tmplen}{1em}
        \begin{tikzpicture}[overlay]
        \tzrectangle[kmcyan,fill=kmcyan]<-5pt,-4pt>
           (0,0)(\tmplen,14pt)
        \tzline[kmcyan]<-5pt,-4pt>(0,0)(\textwidth,0)
        \tznode<.4em,3pt>(0,0){\color{white}\thesection}
        \end{tikzpicture}
        \hspace*{\dimexpr\tmplen+2pt\relax}
    }
    \setsecheadstyle{\sffamily\bfseries\Large}
}
\setsubsechook{
    \setsecnumformat{
       \sffamily\bfseries\large
       \color{kmcyan!50}\thesubsection\quad
    }
}
```

TikZ (tzplot)로 파란 상자를 그리고 섹션 번호를 그 안에 찍은 다음 줄을 그었다. subsection은 간단히 번호의 모양만을 바꾸었다.

pgreenbook은 섹션과 서브섹션의 번호가 판면 밖을 왼쪽으로 벗어나게 디자인되어 있는데 이것은 `\hangsecnum`을 선언한 것이다.

4

목차와 캡션

1 Table of Contents의 이해

이 책의 TOC 디자인에 대해 알아보기 전에 memoir/oblivoir가 이 부분을 어떻게 만드는지 대략 살펴보자. 그림 4.1에 표시된 각 부분별로 설명하고자 한다.

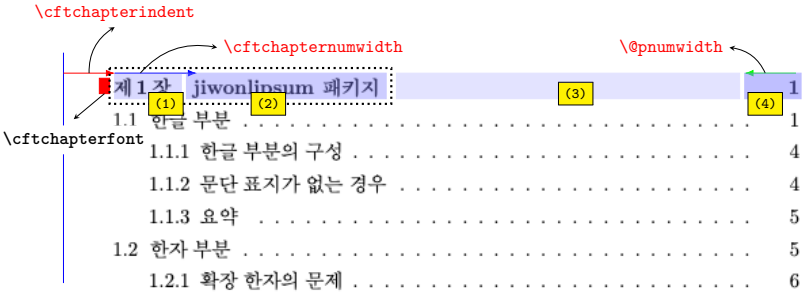


그림 4.1: oblivoir의 toc

1.1 길이

그림 4.1에 표시된 길이 변수는 다음 세 가지가 있다: `\cftKindent`, `\cftKnumwidth`, `\@pnumwidth`. 이 가운데 마지막의 `\@pnumwidth`는 K (=`chapter`, `section`, etc)에 따라 달라지지 않고 일정하기 때문에 이를 바꾸기 위해서는

```
\setpnumwidth{<dim>}
```

으로 설정한다. 그림에는 나와 있지 않지만 페이지 번호가 찍히는 위치가 typeblock의 오른쪽 끝이 아니어야 할 때는 `\@tocrmarg`라는 길이 변수를 조절할 수 있다. 이름 그대로 오른쪽 마진을 추가하는 것이다. 이를 설정하려면

```
\setrmarg{<dim>}
```

이 값은 기본적으로 0pt이다.

chapter, section 등의 수준에 따라서 indent와 numwidth를 별도로 지정할 수 있는데 이들은 길이값이므로 `\setlength`로 변경해도 되지만

```
\cftsetindents{<kind>}{<indent>}{<numwidth>}
```

명령을 쓰는 것이 좋다. `\cftsetindents{chapter}{0pt}{3.5em}` 이렇게 하면 `\cftchapterindent` 0pt, `\cftchapternumwidth` 3.5em 으로 설정한다.

1.2 폰트

\cftchapterfont, \cftsectionfont 등의 매크로는 그 행 전체의 폰트를 지정하는 명령이 된다. 앞으로 설명할 것과 같이 원한다면 각 세부 부분의 폰트를 별도로 지정할 수 있지만 그렇게 하지 않으면 이 매크로가 페이지 번호까지 —\cftKpagefont를 설정하지 않았다면— 영향을 미친다. \renewcommand로 바꿀 수 있다.

1.3 〈num〉 파트

그림에서 (1)로 표시된 부분을 편의상 "〈num〉 파트"라고 부르기로 하자. 즉 chapter에서는 "제 1 장", section에서는 "1.1"에 해당하는 부분이다.

이 부분을 식자하라는 memoir 내부 명령이 \chapternumberline과 \numberline이다. 달리 말하면 chapter 레벨의 \numberline은 \chapternumberline이다. section 이하 레벨에서 \numberline은 그대로 \numberline을 사용한다. 이 명령들은 그 인자로 식자할 〈num〉 파트의 내용(주로 번호)를 받아서 이를 다음과 같이 처리한다.

(i) 먼저 \chapternumberlinehook 또는 \numberlinehook이 정의되어 있다면 이를 실행한다. 당연히 디폴트는 아무 것도 하지 않는 것이다.

(ii) \chapternumberlinebox 또는 \numberlinebox라는 박스를 만들고 이를 식자한다. 이 박스 명령은 두 개의 인자를 취하여, 하나는 박스의 가로 길이이고 다른 하나는 \numberline에서 넘겨받은 "식자할 것"이다.

여기 설명한 매카니즘을 이용하여 〈num〉 파트 식자 방법을 원하는 대로 바꿀 수 있다. 그러나 그런 것은 고급 사용자나 패키지 개발자에게 맡기고, 우리는 디폴트로 주어진 numberline 식자 매카니즘에 익숙해지도록 하자. 개략적으로 말하면, chapter의 numberlinebox의 가로 길이는 당연히 \cftchapternumwidth이고, box의 구성은 대략 다음과 같다. 다음에서 K는 chapter와 같은 레벨의 명칭이다.

\cftKname \cftKpresnum [NUM] \cftKaftersnum \hfill

\cftchaptername은 그 이름과 상관없이 일종의 hook 명령이고 기본값은 아무 것도 하지 않는 것이다. 이 매크로의 영향은 〈num〉 파트에만 미치므로 이를 활용할 일이 있을 수 있다.

주의: section 레벨에서 `\cftsectionpresnum`이 `\hfill`로 되어 있는 것이 기본값임을 주의하라. section의 〈num〉 파트의 정렬을 고려할 때 이 점을 잘 기억해야 한다.

oblivoir의 NUM 영어로 된 책이라면 "Chapter 1"과 같은 〈num〉의 형식에서 "Chapter" 부분은 `\cftchaptername`을 `\chaptername`으로 정의함으로써 구현하고 [NUM]에는 그냥 숫자 "1"만 오면 된다.

그런데 한국어 책에서는 이것이 "제 1 장"하는 식으로 숫자의 앞뒤에 뭔가가 붙는다. oblivoir는 이를 위해서 `\hchaptertitlehead`라는 매크로를 제공하고 있다. 이것은 아예 chapter number에 `\pre@chapter`와 `\post@chapter`를 붙여서 넘겨주는 것이다. (원래 이 매크로는 toc와 page running heading에서 쓰기 위해 고안된 것이었다.) `\hchaptertitlehead`는 chapter style을 정의하면서 `\renewcommand`한다. 그리고 toc에는 이것이 찍힌다. 만약 toc에서 "제"라는 것을 제거하고 싶다면, `\hchaptertitlehead`를

```
\renewcommand*\hchaptertitlehead{\value{chapter}~\post@chapter}
```

으로 미리 정의해두면 된다. 이것은 본문에서 장 번호를 식자할 때는 영향이 없다.

1.4 〈title〉 파트

그 다음으로 (2)의 〈title〉 파트가 온다. 이것은

```
\cftKaftersnumb [TITLE]
```

로 식자되는 간단한 것이다. `\cftchapteraftersnumb`가 [TITLE] 텍스트에 영향을 미치는 매크로임을 알아두는 것으로 충분하다.

그리고 (1)의 〈num〉 파트와 (2)의 〈title〉 파트는 다시 하나의 그룹으로 묶여 있다.

```
<num> part  <title> part
```

그림의 점선이 이를 표시한다.

1.5 〈leaders〉 파트

(3)의 부분은 보통 점을 찍어서 표현하는 부분으로 〈leaders〉 파트라고 부르기로 한다.

이 파트를 식자하는 명령이 \cftKleader이다. 기본적으로 다음과 같이 정의되어 있다.

```
\cftchapterleader = \cftchapterdotsep{\cftnodots}
\cftsectionleader = \normalfont\cftdotfill{\cftsectiondotsep}
```

이 매크로들은 모두 \renewcommand한다. 예컨대 section의 점을 좀더 촘촘하게 하고 싶으면 \cftsectiondotsep의 값을 적절하게 줄여본다. 4.5가 기본이고 단위 없이 숫자를 넣어야 한다. chapter에서와 같이 아무것도 찍지 않으려면 \cftnodots를 이용한다.

1.6 〈pnum〉 파트

(4)는 페이지 번호를 찍는 부분이다. 〈pnum〉 파트라고 부르기로 하자. 이 부분은 하나의 박스인데 그 가로폭은 \@pnumwidth로 정해져 있다. chapter와 section에 따라 다른 폭을 주는 것은 비상식적이므로 그렇게 세분되어 있지 않다.

〈pnum〉 파트는 \cftKformatpnum로 식자한다. chapter라면 인자로서 페이지 번호를 취하는 \cftchapterformatpnum이라는 명령이 여기서 실행된다. 이 매크로는 다시 \cftchapterformatpnumhook이라는 명령을 실행한다. 이 명령은 페이지 번호를 인자로 취하여 페이지 번호를 장식하거나 하는 데 쓰일 수 있다. 페이지 번호를 찍을 때 사용하는 폰트는 \cftKpagefont 매크로이다. \renewcommand할 수 있다.

그러면, 다음처럼 한다면 어떻게 될까?

```
\renewcommand*\cftchapterpagefont{\sffamily\bfseries}
\renewcommand*\cftchapterformatpnumhook[1]{\fbox{#1}}
```

결과를 예측해보자.

1.7 페이지 번호 이후

한 줄의 목록 행을 식자한 후, 마지막에 오는 것은 다음 두 매크로이다.

```
\cftKafterpnum\par
```

보통은 아무 것도 할 게 없지만, 이 매크로로 할 수 있는 이 중에는 다음과 같은 것이 있다. 즉, chapter 목록 행에 대해서, 페이지 번호와 목록 타이틀 사이에 leaders를 두지 않고 쉼표와 페이지 번호를 바로 잇대어 붙이고 싶을 때이다.

```
\renewcommand*\cftchapterleader{, }
\renewcommand*\cftchapterafterpnum{\cftparfillskip}
```

여기 쓰인 \cftparfillskip은 \hskip\parfillskip과 거의 같다.

2 이 책의 TOC

이 책에서는 toc에 특별한 장식을 가하지 않았다. 다음과 같이 적당한 길이 값만을 부여하였다.

```
\cftsetindents{section}{1.3em}{2.5em}
\cftsetindents{subsection}{4em}{2em}
\renewcommand\cftsectiondotsep{3.5}
\renewcommand\cftsubsectiondotsep{3.5}
```

이 책은 본문에서 subsection까지 절번호가 붙고 TOC 목록을 subsection 수준까지 만든다.

```
\maxsecnumdepth{subsection}
\maxtocdepth{subsection}
```

3 캡션

oblivoir에는 [figtabcapt 옵션이 있어서 한국식 캡션 타이틀을 더 쉽게 만들 수 있게 하고 있다. 이에 대해서는 설명서에 자세하므로 이를 참고하라.

이 책은 이 기능에 의지하지 않고 다음과 같이 간단히 처리하고 있다.

```
\captiondelim{\quad}
\captionnamefont{\small\sffamily}
\captiontitlefont{\small\normalfont}
```

지나가는 길에 한 마디 적자면, 플로트가 아닐 때 캡션을 붙이기 위해 \newfixedcaption 명령으로 새로운 캡션 명령을 만들어 쓰는 것이 가능하다.

```
\newfixedcaption{\mycaptionfig}{figure}
```

이제 \begin{figure} 안에 들어 있지 않은 그림에도 \mycaptionfig 명령으로 캡션을 붙일 수 있다.

만약 캡션을 예컨대

[그림 1] **그림의 캡션**

과 같은 모양으로 만들려면 다음과 같이 한다. 일단 `[figtabcapt]` 옵션을 oblivoir에 부여한 후에,

```
\obCaptionFont{\sffamily}
\renewcommand*\obCaptionnameOpen{[}
\renewcommand*\obCaptionnameClose{]}
\captiondelim{\quad}
\captionnamefont{\sffamily}
\captiontitlefont{\normalfont}
```

5

인덱스

1 자동 인덱스 명령

찾아보기는 책의 중요한 구성요소지만 원고를 쓰는 입장에서 어떤 단어를 포함할 것인지 결정하고 일일이 \index 명령을 달아두는 것이 상당히 피곤하다. 이 책에서는 \myem이라는 명령을 정의하고 있다.

```
\myem{우리나라} = 우리나라\index{우리나라}
\myem[우리]{나라} = 나라\index{나라}(우리)\index{우리}
```

이렇게 사용한다.

```
\NewDocumentCommand{\myem}{om}{%
    \IfNoValueTF{#1}%
    {#2\index{#2}}%
    {#2\index{#2}(#1)\index{#1}}}
```

2 printindex 스타일링

theindex 환경은 2단으로 조판되는 것이 기본이다. 이를 바꾸려면 약간 복잡한 방법을 써야 하는데 그 예가 pgreenbook에 나와 있다. (보통은 2단으로 충분하다.)

```
\renewenvironment{theindex}{%
  \renewcommand\indexspace{\vskip\onelineskip}
    \@restonecoltrue
    \setlength{\columnseprule}{\indexrule}%
    \setlength{\columnsep}{\indexcolsep}%
  \chapter*{\indexname}
  \markright{\indexname}
  \preindexhook
  \setlength{\columnsep}{1.8em}%
  \begin{multicols}{3}%
  \linespread{1.2}\small
  \small
  \spaceskip=.275em plus .1em minus .3pt
  \flushcolumns
  \indexmark
    \phantomsection
    \addcontentsline{toc}{chapter}{\indexname}%
    \parindent\z@
  \parskip\z@ \@plus .3\p@\relax
  \let\item\@idxitem}%
 {\end{multicols}
}

\renewcommand{\@idxitem}{\par\raggedright\hangindent 20\p@}
```

찾아보기

그림 5.1: pgreenbook의 찾아보기 스타일

multicol 패키지를 이용하여 3단 조판하였다. pgreenbooksample 문서의 찾아보기는 그림 5.1과 같은 모양이다.

3 인덱스 처리 명령

인덱스의 생성은 외부 유틸리티에 의한다. 한글 인덱스를 처리할 수 있는 것은 현재 komkindex, xindy (texindy), xindex이다. 이 가운데 xindex는 개발 중이라고 볼 수 있고 전통적으로 komkindex를 가장 많이 사용한다. imakeidx라고 인덱스 생성을 자동화할 수 있는 패키지가 있으나 makeindex만을 지원하고 komkindex를 쓸 수 없기 때문에 texindy에만 적용 가능하다. 이 패키지에 대해서는 더 언급하지 않겠다.

```
komkindex -s kotex -k <main>.idx
```

`-k` 옵션은 영문 단어보다 한글 단어가 먼저 나오게 정렬하라는 옵션이다.

`-s` 옵션은 `kotex.ist`라는 스타일 정의 파일을 적용하라는 뜻이다. 만약 다른 ist 파일이 있다면 파일 이름을 적어주면 된다. `kotex.ist`로 생성하는 인덱스는 lshort-ko의 인덱스와 모양이 같다.

다음은 kotex.ist의 내용이다.

```
preamble         "\\begin{theindex}\n
\\def\\hindexhead#1{\\ifcase#1ㄱ\\or ㄴ\\or ㄷ\\or ㄹ\\or
  ㅁ\\or ㅂ\\or ㅅ\\or ㅿ\\or ㅇ\\or ㅇ\\or ㅈ\\or ㅊ\\or ㅋ\\or
```

```
  ㅌ\\or ㅍ\\or ㅎ\\or ㅗ\\or ㅏ\\or ㅑ\\or ㅓ\\or ㅕ\\or ㅛ\\or
  ㅠ\\or ㅜ\\or ㅠ\\or ㅡ\\or ㅣ\\or ㆍ\\else 종성\\fi}\n\n"

headings_flag       1
group_skip          "\n\n\\indexspace\n"
item_0              "\n\\item "
heading_prefix      "【 "
heading_suffix      " 】\\nopagebreak\n"
symhead_positive    "기호"
numhead_positive    "숫자"
delim_0             "\\dotfill "
delim_1             "\\dotfill "
delim_2             "\\dotfill "
```

자신의 ist를 만들어야 한다면 이것을 기준으로 조금씩 수정하는 것이 좋다. 예컨대 ksminitex과 함께 배포되는 ksminitex.ist는 다음과 같이 되어 있는데, 이것은 kotex.ist에서 점선 부분을 없앤 것이다.

```
preamble "\\begin{theindex}\n
\\def\\hindexhead#1{\\ifcase#1가\\or 나\\or 다\\or 라\\or
  마\\or 바\\or 사\\or ㅇ\\or 아\\or ㅇ\\or 자\\or 차\\or 카\\or
  타\\or 파\\or 하\\or ㅗ\\or ㅏ\\or ㅑ\\or ㅓ\\or ㅕ\\or ㅛ\\or
  ㅠ\\or ㅜ\\or ㅠ\\or ㅡ\\or ㅣ\\or ㆍ\\else 종성\\fi}\n\n"

headings_flag     1
group_skip          "\n\n\\indexspace\n"
item_0              "\n\\item "
heading_prefix      "【 "
heading_suffix      " 】\\nopagebreak\n"
symhead_positive    "기호"
numhead_positive    "숫자"
delim_0             ", "
delim_1             ", "
delim_2             ", "
```

pgreenbook은 `ind` 생성에 xindy를 이용하였다. 다음 명령으로 실행한다.

```
texindy -C utf-8 -L korean <main>.idx
```

6

정리류

1 정리, 명제, …

수학식과는 달리 텍스트(와 수식)로 이루어져 있지만 수학적으로 특별한 의미를 지니는 문단을 '정리류 문단(theorem-like paragraphs)'이라 한다. 명제(proposition), 정리(theorem), 정의(definition), 보조정리(lemma), 따름정리(corollary), 주의(remark) 등이 여기에 속하며 증명(proof)도 특별한 정리류 문단이다.

일반적인 논문이라면 다음과 같은 amsthm이 제공하는 정리류 문단이 적절하겠으나,

Theorem 1 (중심극한정리). 서로 독립이며 동일한 분포를 따르는 확률변수 $X_1, X_2, \cdots, X_n$에 대해, 각각의 평균은 $E(X_i) = \mu$이고 각각의 표준편차는 σ라 하자. $\xi_n = \dfrac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$ 라 둘 때, ξ_n은 표준정규분포로 분포수렴한다.

단행본에서는 이렇게 해서는 시각적으로 잘 눈에 들어오지 않기 때문에(특히 한글 책에서) 박스를 치거나 문단에 장식을 하여 두드러지게 만드는 관행이 있다. 이를 위하여 tcolorbox의 `theorems` 라이브러리를 활용하는 경우가 많다.

정리 1.1 (중심극한정리)
서로 독립이며 동일한 분포를 따르는 확률변수 $X_1, X_2, \cdots, X_n$에 대해, 각각의 평균은 $E(X_i) = \mu$이고 각각의 표준편차는 σ라 하자. $\xi_n = \dfrac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$ 라 둘 때, ξ_n은 표준정규분포로 분포수렴한다.

이 환경을 corollary나 lemma로 확장할 수 있다.

2 박스 디자인에 대한 코멘트

pgreenbook은 이들 박스를 아주 심플한 framed와 shaded만으로 구현하였다.

다른 책의 디자인을 보면 더 복잡하고 예쁜 정리류 박스 문단을 심심찮게 발견할 수 있는데, LaTeX에서는 tcolorbox 패키지나 tikz를 이용하여 얼마든지 원하는 박스형식의 정리류 문단을 만들 수 있다. 심지어 tcolorbox 패키지의 부속 라이브러리 중에는 theorem이라는 것도 있다.

항상 그렇지만, 과도한 박스형 문단의 사용은 골치아픈 문제를 수반한다. 가장 대표적인 것이 "잘라지는 박스"의 문제이다. 박스가 잘라지지 않으면 페이지의 아랫쪽이 휑하니 비어서 곤란한 경우가 생기지만 막상 이것을 분질

러놓으면(tcolorbox로는 잘라지는 박스를 만드는 것이 어렵지 않다) 어떻게 해도 안 예쁜 결정적인 문제가 생긴다.

그리고 박스 안의 내용보다 박스 형태에 더 눈길이 가는 디자인은 곤란하다. 박스는 거의 신경쓰이지 않을 정도로, 단지 다른 문단과 구별이 되도록 하는 정도의 자기 주장이 강하지 않은 디자인이 좋은 디자인이다.

3 counter의 종속

책을 만들 때 중요한 카운터를 열거하면

page, chapter, section, subsection, figure, table, equation

카운터에 대하여 변경을 가하는 명령으로

```
\setcounter{<counter>}{<number>}
\stepcounter{<counter>}
\refstepcounter{<counter>}
\addtocounter{<counter>}{<number>}
```

이 정도를 숙지해두면 될 것이다. `\refstepcounter`는 `\stepcounter`와 같지만 `\label`을 붙였을 때 이 번호에 대하여 refer할 수 있게 해주는 특별한 명령이다.

예컨대 현재 페이지를 1000으로 바꾸려면

```
\setcounter{page}{1000}
```

LaTeX에서 counter는 다음 두 명령에 의하여 제어한다.

```
\counterwithin
\counterwithout
```

예컨대 section 번호는 2.3과 같이 chapter 번호에 종속되어 있다. 이것은 (기본값이기는 하지만)

```
\counterwithin{section}{chapter}
```

와 같이 한 것이다. 그러므로

```
\counterwithout{section}{chapter}
```

라고 하면 단지 절 번호만이 표시된다. 그런데 문제는 "표현 형태"에서는 chapter에 종속되지 않게 하고 싶으나 chapter가 갱신될 때 section도 리셋되게 하면 좋겠다. 이럴 때는

```
\counterwithout{section}{chapter}
\counterwithin*{section}{chapter}
```

이와 같이 별표붙은 명령(카운터를 종속시키되 표현형태는 바꾸지 말라는 의미)을 쓰면 되겠다.

equation이나 figure의 카운터도 같은 방식으로 할 수 있다. 그런데 theorem, example, remark 카운터는 그 카운터 이름을 사용자가 정할 수 있게 되어 있으므로 주의하도록 한다.