도은이아빠

hangul-k 사용 설명서

차 례

그림 차례

표 차례

제 1 장

간단한 설치와 사용법

제 1 절 문서작성의 간단한 방법

HLaTeX으로 문서를 작성하는 두 가지 방법이 있다. 하나는 한글을 "표시"만 하면서 모든 서식은 영문 문서의 틀을 그대로 따르는 것으로, hfont 패키지를 이용한다.

```
\usepackage{hfont}
```

다른 하나는 한글 서식을 한글 환경에 맞도록 설정해주는 것으로, 이 때는 장과 절의 모양, heading의 모양, 조사의 처리 등 "문서 틀잡기"를 한글 문서화할 수 있다. HLaTeX의 한글 서식은 hangul 패키지였다.

```
\usepackage{hangul}
```

hangul-k 패키지는 hangul 패키지를 대체하려는 것이다. 일반적인 상황에서는 hangul이 쓰여왔지만, 특히 PDF 파일을 만들면서 hyperref을 이용

하여 하이퍼링크를 넣으려 하는 경우 바람직하지 못한 결과가 만들어지곤 했기 때문이다.

그러므로, 한글 PDF 문서는 다음 그림 1.1과 같이 시작한다.

```
\documentclass{article}
\usepackage{hangul-k}
\usepackage[dvipdfm,bookmarks=false,colorlinks]{hyperref}
```

그림 1.1: 한글 PDF 문서 작성 예제

제 2 절 설치

2.1 hangul-k 패키지 설치

hangul-k 패키지를 설치하는 것은 다음 절차를 따른다. 여기서는 윈도 XP에서 MiKTEX 사용자의 경우를 예로 든다. 다른 TEX 배포판을 사용하고 있다면 유사한 방법으로 설치하는 것이 어렵지 않을 것이다.

hangul-k 패키지를 사용하려면 MiKTEX과 같은 TEX 배포판[1]과 HLATEX 0.991 이상이 필요하다. 이 시스템의 설치에 대해서는 KTUG Faq 사이트를 참고하라.

(i) 다운로드 hangul-k 패키지 홈페이지에 접속하여 최신 버전을 확인한다. 최신 버전은 `.zip` 압축파일로 제공된다.

Microsoft Internet Explorer라면 이 파일이름의 링크 위에서 오른쪽 마우스 버튼을 눌러서 "다른 이름으로 저장"을 선택하여 적당한 곳에 저장한다.(그림 1.2 참조)

다운로드 받은 곳이 [내 문서] 내의 [받은 파일] 폴더라고 하자.

[1] 참고. http://faq.ktug.or.kr/mywiki/TeXImplementations.

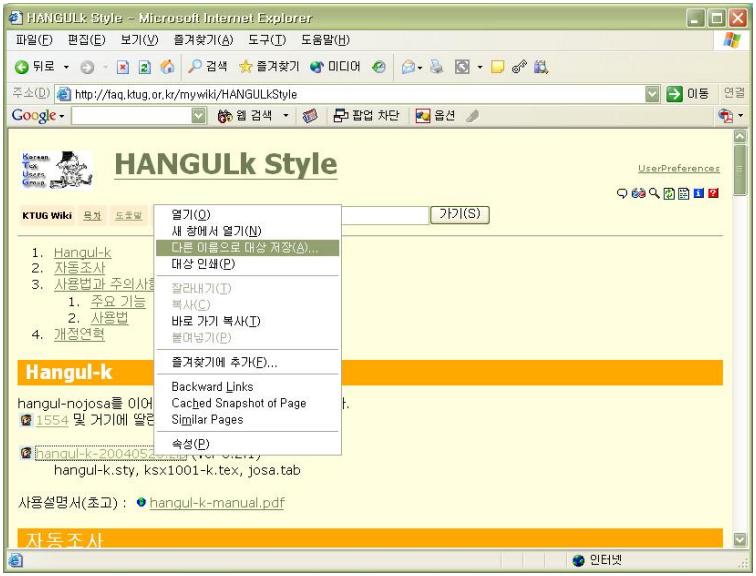


그림 1.2: Internet Explorer로 파일 다운로드

이 폴더를 열면 `hangul-k-20040526.zip` 파일이 저장되어 있을 것이다. 이 파일을 풀면 `hangul-k.sty`, `josa.tab`, `ksx1001-k.tex` 세 개의 파일을 얻을 수 있다.

(ii) 파일 위치 복사 사용자의 Local TEXMF tree가 `C:\localtexmf`라고 하자. 이 경우,

`C:\localtexmf\tex\latex\hlatex\contrib`

라는 폴더를 하나 만들고, 위의 `hangul-k.sty`과 `ksx1001-k.tex`을 이곳으로 복사한다.

`josa.tab`은 HLATEX이 설치된 위치에 있는 `josa.tab`을 교체하여야 한다. MiKTEX-KTUG 2.3으로 HLATEX을 사용하고 있다면 이 파일은

`C:\texmf\tex\latex\hlatex`

폴더에 이미 있을 것이다. 기존의 파일을 이름을 바꾸든지 해서 백업해두고 다운로드받은 `josa.tab`으로 교체한다.

(iii) Filename Database 갱신 이제 TEX 시스템의 Filename Database를 갱신해준다. MiKTeX Options를 실행하여 ‘‘`Refresh Now`’’ 버튼을 눌러도 되고, 명령행을 열어서

```
#> initexmf -u
```

와 같은 명령을 이용해도 된다.[2]

2.2 은글꼴 설치

은글꼴은 박원규 님이 HLATEX의 UHC 글꼴을 TrueType으로 포팅한 것이다. hangul-k 패키지와 DHHangul에서 필요하므로 설치를 권장한다.

HLATEX을 위한 은글꼴 설정은 김도현 님에 의하여 TEXMF TREE 압축파일 형태로 준비되어 있다. KTUG FAQ 은글꼴 페이지에서 다운로드받을 수 있다(약 53메가 가량). 이 통합 TEXMF TREE 압축파일을 다운로드 받아서 적당한 곳에 풀어놓은 다음, MiKTEX이라면 MiKTeX Options를 실행하여 `[ROOTS]` 탭에서 등록하고 Filename Database를 갱신하여 바로 사용할 수 있다.[3]

[2] 일반적인 TEX 배포판에서는 `mktexlsr` 명령이 더 흔하게 쓰일 것이다.

[3] MiKTEX이 아니라 teTeX 등 Web2C 계열의 TEX 배포판이라면, `texmf.cnf`에 등록함으로써 같은 결과를 얻을 수 있다. 다만 W32TEX과 같이 New TDS를 채택한 일부 TEX 배포판들의 경우에는 `.map` 파일을 적절한 위치로 옮겨 주어야 할 것이다. 자세한 사항은 KTUG FAQ나 각 배포판의 도움말을 참고하라.

제 2 장

한글 문서 작성하기

이 단원은 은광희 지음, 『HLaTeX 길잡이』[은광희2000]의 내용을 바탕으로 hangul-k 패키지에 해당하는 사항을 보충하였다.

제 1 절 한글 문서의 틀

1.1 문서의 기본틀과 문서클래스

LaTeX 2ε 문서는

```
\documentclass[옵션인자]{문서클래스}
```

로 시작한다.[1] 그림 2.1은 [은광희2000]에 있는 것을 조금 수정한 것으로, 문서의 기본 모양을 잘 보여준다.

문서클래스는 문서의 바탕을 지정하는 것으로 LaTeX 2ε에서 제공되는 article, book, report, letter 등을 말한다. 학회나 기타 다른 목적으로 제작

[1] LaTeX 2.09에서는 `\documentstyle`이라는 명령을 사용하였다.

```
\documentclass[옵션인자]{클래스}

\usepackage[옵션인자]{hangul-k}  ← hangul-k 꾸러미 대신 hfont 또는 hangul 꾸러미를 쓸 수 있다.
\usepackage[옵션인자]{패키지}    ← 필요에 따라 사용될 각종 꾸러미를 지정한다.
\usepackage[dvipdfm,             ← hangul-k는 hyperref 패키지를
bookmarks=false]{hyperref}         필요로 한다.
...                              ← 꾸러미뿐만 아니라 문자를 출력하지 않는 다른 명령들을 여기에서 선언할 수 있다.
\begin{document}                 ← 이 전까지를 전문(前文: preamble)이라고 한다.
나라의 말이 중국과 달라서       ← 문서의 작성은 TEX의 관습에
한자로는 서로 통하지 아니하므로   따른다. 즉, 중첩된 공간
이런 까닭으로 어리석은 백성들이   문자는 하나로 처리된다.
말하고자 하는 바가 있어도       복귀 문자가 한번 나오면
마침내 제뜻을 능히 펴지 못하는   공간 문자로 처리되나, 두번
사람이 많으니라. 내가 이를       이상 중첩되면 하나의 복귀
불쌍히 여겨 새로 스물여덟자를     문자 역할을 한다... 등등.
만드나니
\end{document}                   ← 이 명령이 나온 후의 모든 문장은 무시된다.
```

그림 2.1: LaTeX 문서의 기본 구조

된 다른 클래스도 있을 수 있으며, 그럴 경우에는 .cls라는 확장명을 갖는 파일로 제공된다.

옵션 인자는 문서의 모양과 같은 선택적 기능을 지시하기 위한 것으로, 클래스에 따라 다양한 옵션이 지정될 수 있다. 예를 들면 a4paper와 같은 종이크기, 11pt와 같은 글자크기 등이 옵션 인자로 올 수 있다. 여러 개의 옵션 인자들은 쉼표(,)로 분리된다.

문서 클래스의 선택과 옵션 인자의 설정에 대해서는 일반 영문 LaTeX을 사용하는 것과 동일하므로, LaTeX 사용법 참고서나 인터넷 문서를 참고하라.

1.2 hangul-k 패키지의 선택사항

한글 사용을 위해서 다음을 선언한다.

```
\usepackage[옵션인자]{hangul-k}
```

hangul-k 패키지의 옵션 인자는 다음과 같은 것이 있다.

[hardbold] 또는 [softbold] [softbold]는 기본 굵기의 폰트를 세 번 어긋나게 겹쳐찍어서 굵은 글자를 표현하는 방법(이른바 Poorman's Bold)이지만, PDF를 만드는 상황에서는 권장되지 않는다. 이 옵션은 HLaTeX hangul과의 호환성을 위해서 남겨 둔 것이다.

[hanja] HLaTeX의 선택사항으로서, 문서의 단원명이나 날짜 등을 한자로 식자하게 한다.

HLaTeX의 [nojosa] 옵션은 hangul-k 패키지에서는 무의미하므로 제외되었다. 따라서, 일반적인 한글 문서를 작성하고 특별히 한자를 문서의 단원명칭이나 날짜 등에 사용할 필요가 없다면, 아래와 같이 선언하는 것으로 충분할 것이다.

```
\usepackage{hangul-k}
```

만약 자동조사 기능을 전혀 사용하지 않는 것이 확실하다면 비록 obsolete 상태이기는 하나 hangul-nojosa를 쓰는 방법도 있다.

1.3 hyperref 선택사항

hyperref 패키지에는 수많은 선택사항과 설정옵션이 있다. 자세한 내용은 hyperref의 패키지 문서와 매뉴얼을 참고하라. 여기서는 hangul-k 패키지와 관련하여 필요한 사항만 지적하려 한다.

hangul-k 패키지는 hyperref의 [dvips], [dvipdfm], [pdftex] 드라이버 옵션을 지원한다. 즉, 이 세 가지 가운데 한 가지 옵션을 명시적으로 지시하여야 자동조사 기능이 작동한다. 그러므로 hangul-k 패키지만을 지정하고 hyperref을 올리지 않으면 에러가 발생할 수 있다.

DVIPDFMx를 이용한 한글 PDF 문서. 한글 트루타입 폰트와 DVIPDFMx를 이용하면 한글 책갈피(bookmarks)를 만들어넣을 수 있고 "한글 문자열이 검색·추출 가능한 PDF"를 만들 수 있다. 한글 트루타입 폰트는 은글꼴이나 그밖의 다른 TrueType 글꼴 세트를 사용한다.

은글꼴 통합 패키지 설치에 대해서는 4페이지의 2.2를 참고하라. 은글꼴은 GNU GPL로 배포되는 유일한 한글 TrueType 글꼴이다.

DVIPDFMx를 위해서는 [dvipdfm] 옵션을 선언한다. hyperref이 아직까지 [dvipdfmx] 옵션을 지원하지 않기 때문이다. [CJKbookmarks] 옵션은 한글 책갈피를 만들기 위해 필요하다. 아래 예제는 한글 책갈피 만들기를 선택하고 은글꼴을 기본 글꼴로 쓰는 경우 문서의 시작 부분 예시이다.

```
\documentclass{article}
\usepackage{hangul-k}
\usepackage[dvipdfm,CJKbookmarks,colorlinks]{hyperref}
\AtBeginDvi{\special{pdf: tounicode KSCms-UHC-UCS2}}
  % 한글 책갈피를 위한 dvipdfmx 설정.
\usepackage{unttf}
```

표 2.1: 한글 PDF를 제작하는 방법

	한글 표현	검색·추출	한글책갈피	PSTricks
DVIPDFMx	가능	가능	가능	불가능
PDFLaTeX	가능	불가능	불가능	불가능
GhostScript	가능	불가능	불가능	가능

PDFLaTeX을 이용하는 경우. PDFLaTeX은 PDF 제작에 관한 한 매우 훌륭한 프로그램이다. 안타깝게도 한글 관련 여러 가지 설정들이 충분히 개발되어 있지 않아서 훌륭한 기능들을 다 활용하지 못하지만 적어도 한글을 "표현"은 할 수 있다. 표 2.1에서 보듯이, 이 경우 한글 책갈피를 만들지 못한다. 책갈피가 반드시 필요하다면 영문 책갈피를 만들어 넣는 방법이 있다. 이 때는 bmsec을 써서 장절명령에서 책갈피에 들어갈 텍스트를 지정해 준다. 예를 들면,

```
\section(Hangul Document){한글 문서 작성}
```

과 같은 방식이다. 장절마다 일일이 이것을 지정해주는 것이 쉽지만은 않다.

일반적으로는 책갈피 만드는 기능을 끄도록 한다. 그래서 문서의 시작 부분은 다음과 같이 된다.

```
\documentclass{article}
\usepackage{hangul-k}
\usepackage[pdftex,bookmarks=false]{hyperref}
```

PDFLaTeX 실행시에는 옵션을 선택적으로 변화시키도록 작성하려면 그림 2.2와 같이 작성한다. 이 설정은 만약 PDFLaTeX이 실행되면 책갈피를 만들지 말고, 그렇지 않으면 DVIPDFMx 용 한글 책갈피를 설정하도록 한다.

dvips와 GhostScript를 이용하는 경우. dvips와 GhostScript의 ps2pdf 스크립트를 이용하여 PDF를 만드는 경우가 있을 수 있다. 특히 pstricks를 이

```
\documentclass{article}
\usepackage{ifpdf}
\usepackage{hangul-k}
\ifpdf
    \usepackage[pdftex,bookmarks=false]{hyperref}
    \usepackage[pdftex]{graphicx}
\else
    \usepackage[dvipdfm,CJKbookmarks]{hyperref}
    \AtBeginDvi{\special{pdf: tounicode KSCms-UHC-UCS2}}
    \usepackage[dvipdfm]{graphicx}
\fi
```

그림 2.2: Preamble 설정의 예시

용한 그림을 포함한 경우이든가, 아니면 seminar 패키지나 prosper를 이용하여 Presentation을 만드는 경우라면 이 방법이 거의 유일한 해결책이다.[2]

hangul-k 패키지와 hyperref을 이용하는 문서는 dvips를 잘 지원하여 하이퍼링크를 만들어준다. 그러나 물론 한글 책갈피나 텍스트의 검색·추출은 현실적으로 구현되기 어려운 상황이다.

이 경우의 문서 시작 부분은 다음과 같을 것이다.

```
\documentclass{article}
\usepackage{hangul-k}
\usepackage[dvips,bookmarks=false]{hyperref}
```

제 2 절 한글 문서의 글자체 설정하기

이 절의 주요 부분은 [은광희2000]을 가져온 것이다. 해당 부분을 hangul-k 패키지에 맞추어서 수정하면서, Lambda(Λ)에 관한 부분은 제외하고,[3] 문체

[2] DVIPDFMx는 MetaPost를 잘 지원한다. 그러므로 DVIPDFMx를 쓰는 것이 더 좋으신 분은 MetaPost에 관심을 가져보시는 것은 어떨는지...

[3] Lambda와 hangul-k 패키지의 관계에 대해서는 제 3 장을 참고하라. 한편, 김도현 님

는 경어체를 평어체로 바꾸었다. 일부 우리말 전산 용어는 원본의 느낌을 전하기 위하여 그대로 두었다.

2.1 글자체 선택

문서를 작성할 때, 어떤 특정한 부분은 현재 쓰이고 있는 글자체와 구분을 해야 하는 경우가 생긴다. 글자의 두께를 진하게 한다든지, 크기를 크게 한다든지, 다른 모양을 쓴다든지, 또는 설계(design)가 전혀 다른 글자체를 쓴다든지 등등.

이를 위해 LaTeX은 여러가지 글자체 바꿈 매크로(macro commands)들을 제공한다. hangul과 hangul-k 패키지는 영문 글자체 바꿈 명령과 한글 글자체 바꿈 명령을 별도로 정의한 다음, 필요에 따라 호환해서 쓸 수 있도록 해두고 있다. 예컨대 `\sffamily`는 영문의 산세리프 글꼴을 선택하는 명령이면서 동시에 한글의 `\gtdefault`로 정의된 글꼴을 선택하는 명령으로도 쓰이도록 되어 있는 것이다. 다만 이러한 호환은 세 가지 기본 글자체에 대해서만 작동한다(rm, sf, tt).

영문 글자체 바꿈 매크로는 NFSS를 이용하고, 한글은 이와 호환되는 HFSS를 별도로 정의하였다. 이들은 모두 LaTeX 2_ε에서 작동한다.[4]

의 DHHangul은 Lambda(Λ)용 한글 TeX 시스템이다. DHHangul 사용자 설명서가 하루빨리 나오기를 바란다.

[4] LaTeX209에서 제공했던 `\rm`, `\sf`, `\sl`, `\it` 등등의 글자체 바꿈 매크로들은 모두가 같은 차원에서 글자체를 바꾸기 때문에 여러가지 단점을 가지고 있었다. 즉, 글자체의 설계가 `\rm`과 다른 `\sf`를 계속 써 오다가 그 글자체의 기울인 모양인 `\sl`로 글자체를 바꾸면 `\rm`의 `\sl`이 되는데, 이는 일단 글자체를 애초값으로 복귀시킨 후 글자체가 바뀌어지도록 되어 있기 때문이었다. 이러한 단점을 보완하기 위해 1989년부터 Frank Mittelbach와 Rainer Schöpf는 "새로운 글자체 선택 방식"(NFSS: New Font Selection Scheme)을 고안하기 시작하여 현재 LaTeX 2_ε에 채택되어 있는 NFSS를 만들었다.

2.1.1 영문 글자체 선택 방법(NFSS)

NFSS는 글자체를 다섯가지 요소로 구분하여 이 요소들을 각각 독립적으로 선택할 수 있게 한다. 그 다섯가지 요소는 그림 2.3과 같다.[5]

명령형과 선언형. 글자체 선택에는 명령형과 선언형이 있다. 명령형이란 예컨대 \textbf{...}와 같이 한 가지 (이상의) 매개변수를 요구하며, 선언형은 \bfseries와 같이 변수를 요구하지 않는다. 두꺼운 글자체를 얻는 위의 명령 또는 선언에서 명령의 효력은 \textbf 명령형의 경우 매개변수로 주어진 텍스트에 대해서만 작용하지만 선언형으로 \bfseries와 같이 선언하면 그 이후에 오는 모든 텍스트에 영향을 미친다. 그러므로 보통 글꼴로 되돌아오기 위해서는 \normalfont와 같이 다시 선언해주어야 한다. 이러한 불편을 피하기 위해서 선언형을 다음과 같이 환경 형태로 쓰기도 한다.

```
몇몇 단어들은
\begin{bfseries}
두껍게
\end{bfseries}
인쇄됩니다.
```

몇몇 단어들은 **두껍게** 인쇄됩니다.

기본 글자체 가족. 문서의 애초(기본) 글자체는 \textnormal 명령 또는 \normalfont 선언으로 선택된다. 이 명령 또는 선언은 문서의 글자체를 초기화시킴으로써 항상 같은 모양을 선택할 수 있게 한다.

문서의 글자체 가족은 로만(roman) 글자체와 산세리프(sans serif) 글자체 그리고 타자(typewriter) 글자체가 있다. 이들을 선택하는 명령형은 다음과 같다:

```
\textrm \textsf \texttt
```

같은 효과를 다음의 선언형으로 얻을 수 있다.

[5] 이 내용은 「한글LaTeX 길잡이」([은광희2000])를 그대로 옮긴 것이다.

글자체 부호화(encoding) 다른 알파벳을 쓸 때 각 나라별로 codepage가 정해져 있습니다. LaTeX에서는 다른 프로그램과 마찬가지로 각 글자와 0–255 사이의 글자체 위치를 연관시킴으로써 이를 해결합니다. LaTeX에서는 이를 "부호화 방법"으로 명합니다.

글자체 가족(family) 한 글자체 가족은 공통의 설계 원칙을 갖습니다. 가족내에서는 크기(size)와 무게(weight), 폭(width), 모양(shape)으로 구성원이 구별됩니다.

글자체 모양(shape) 글자체 가족의 구성원을 구별하는 중요한 속성으로서 "곧은(upright) 모양"과 "이탤릭*(italic)* 모양" "기울인*(slanted)* 모양" 혹은 대문자의 크기를 70%정도로 축소하여 소문자로 쓰는 "작은키(Small Caps) 모양"등으로 구별됩니다. 글자체 모양에 있어서 그 중요도는 떨어지지만 "외곽형(outlined) 모양"과 "그림자(shaded) 모양" 및 "곧은 이탤릭(upright italic) 모양" 등도 글자체 모양에 속합니다.

글자체 무게(weight) 및 폭(width) 무게와 폭은 서로의 조합으로 하나의 글자체 속성을 형성합니다. 무게는 글자의 두께로서 결정되며 "보통 두께(medium)", "가벼운 두께(light)", "무거운 두께(bold)" ... 등으로 나뉩니다. "가벼운 두께"에도 "아주 가벼운 두께(extra light)", "매우 가벼운 두께(ultra light)", "보통 가벼운 두께(light)", "약간 가벼운 두께(semi light)" 등등으로 분리되며 "두꺼운 두께"도 이런 식으로 세분됩니다. 폭은 "보통의 폭(medium width)", "확장된 폭(extended width)"등이 있습니다.

글자체 크기(size) 글자체 크기는 인쇄기의 점을 단위로 표현됩니다. 인쇄기의 점은 TeX에서 pt라는 단위를 쓰며 1인치는 72.27pt입니다. LaTeX에서는 글자체 크기가 1.2의 승으로 구별됩니다. 이렇게 정한 배경에는 후에 축소 복사를 할 때의 편이함이 들어있습니다. 즉 A5 크기의 책자를 만들 경우, A4 종이에 $1.44 = \sqrt{2}$의 글자체 크기를 선택합니다.

그림 2.3: NFSS의 글자체 구성요소

```
\rmfamily \sffamily \ttfamily
```

로만 글자체는 일반적으로 문서의 기본 글자체로 선언되어 있다.

글자체 시리즈(series). 문서의 글자체 시리즈(series)는 폭과 두께의 두 요소로 구성되며, 다음과 같은 명령 및 선언형이 있다. 애초값은 `\textbf`가 확장된 두꺼운 두께(bold extended)이고 `\textmd`는 보통의 폭과 보통의 두께이다.

명령형	선언형
`\textmd`	`\mdseries`
`\textbf`	`\bfseries`

글자체 모양(shape). 글자체 모양은 애초값인 곧은(upright) 모양을 비롯하여 이탤릭(*italic*) 모양, 작은키(SMALL CAPITAL) 모양, 기울인(*slanted*) 모양이 있는데 이들을 선택하는 명령과 선언형은 다음과 같다. 이 가운데 이탤릭 모양과 곧은 모양 사이의 전환은 `\emph` 명령이나 `\em` 선언으로 이루어진다.

명령형	선언형
`\textup`	`\upshape`
`\textit`	`\itshape`
`\textsc`	`\scshape`
`\textsl`	`\slshape`

글자체 크기(size). 글자체 크기는 10가지의 선언형만 있다. 명령형은 없다. 선언형의 형태와 실제 예는 그림 2.4를 보라.

글자체 관련 내부 명령들. 위와 같은 외형 글자체 바꿈 명령들의 행동 방식은 내형 글자체 바꿈 명령에 의해 달라진다. 예를 들어

```
\renewcommand{\familydefault}{cmss}
```

명령(선언형)	보기
\tiny	미소한 크기
\scriptsize	각본철 크기
\footnotesize	주석 크기
\small	작은 크기
\normalsize	보통 크기
\large	큰 크기
\Large	좀더 큰 크기
\LARGE	아주 큰 크기
\huge	거대한 크기
\Huge	아주 거대한 크기

그림 2.4: 글자체 크기

와 같은 내형 글자체 바꿈 명령을 쓰면 문서의 주 글자체 가족이 cmss(Computer Modern Sans)로 바뀜으로써 주석 및 머릿말 등, 외형 글자체 바꿈 명령만으로는 바뀌지 않는 부분까지의 글자체 가족이 영향을 받게 된다.

LaTeX에서 애초값은 표 2.2와 같이 설정되어 있다.

2.2 우리말 글자체 선택: 개요

사용자의 입장에서 글자체(글꼴)를 바꾸고 선택하는 방법을 간단하게 정리하면 다음과 같다.

글꼴 가족. 기본 글꼴 가족 선언으로 \mjfamily, \gtfamily, \tzfamily가 정의되어 있고, 이 각각은 영문 글자체 선택 방법의 \rmfamily, \sffamily, \ttfamily에 대응한다. 이 세 가지 기본 글꼴에 대해서는 어떤 선언형 매크로를 써도 된다. 다만 한글형 선언은 한글 텍스트에만 영향을 미칠 것이다. 영문형 선언은 영문과 한글 모두를 바꾸어준다. 명령형은 \textmj, \textgt, \texttz가 있으며 역시 영문 명령형 \textrm, \textsf, \texttt에

표 2.2: LaTeX의 글자체 기본값

내형 글자체	애초값
\encodingdefault	OT1
\familydefault	\rmdefault
\seriesdefault	m
\shapedefault	n
\rmdefault	cmr
\sfdefault	cmss
\ttdefault	cmtt
\bfdefault	bx
\mddefault	m
\itdefault	it
\sldefault	sl
\scdefault	sc
\updefault	n

각각 대응한다.

그 이외의 글꼴 가족은 \hfontfamily[6]매크로를 이용하여 선언한다. 예컨대 신문 글꼴을 지정하려면 \hfontfamily{sh}와 같은 방법을 쓰는 것이 좋다.[7]

하나의 글꼴 가족을 구성하는 방법과 추가 글꼴 가족의 선택에 대해서는 HFSS 관련 2.3 절과, 표 2.4를 보라.

글꼴 시리즈. 한글 글꼴 가족에서는 주로 b, bx 시리즈들이 사용되는 경우가 많다. bx 시리즈가 별도로 지정되어 있지 않은 경우에는 b 시리즈를 그대로 가져다 쓴다.[8]

[6] Lambda에서는 \fontfamily

[7] TTF2HLaTeXFont 스크립트를 이용하여 사용자가 직접 만든 글꼴 세트에서는 \textsh와 같은 형식의 명령형도 함께 사용할 수 있도록 정의해주고 있다.

[8] 이 때문에 은글꼴을 사용하는 경우 이따금 컴파일할 때 "모든 글꼴 속성을 다 지원하지 못했다"는 경고(warning)가 로그에 나오지만, 큰 문제가 되지 않으므로 무시하면 될

두꺼운 글꼴을 선택하려면 \textbf 명령이나 \bfseries 선언을 이용한다. 일반적인 글꼴 시리즈 선택을 위해서는 \hfontseries 명령이 있다.

기울인 글꼴. \emph 명령을 쓰거나 \em 선언을 쓰는 경우 HLaTeX은 기울인 글꼴을 식자해주는 것이 기본값이다. 76페이지를 보면 이 기울인 글꼴 문제에 대한 언급이 있다.

다음 소절에서는 HLaTeX의 글자체 선택방법을 좀더 자세히 알아본다.

2.3 우리말 글자체 선택 방법: HFSS

HFSS는 NFSS를 바탕으로 하고 있고 우리말 부호화의 특성에 따라 새로운 \hfontencoding과 \hfontfamily가 추가 되었으며 글자체 선택 방식도 우리말 글자체에 맞도록 변경되었다.

글자체 부호화 우리말의 부호화는 H로 설정되어 있고 NFSS에서 사용되는 글자체 부호화 설정 매크로 명령 \fontencoding을 우리말화한 \hfontencoding을 쓴다. 우리말 글자체의 부호화는 사용자의 입장에서 볼 때 달리 설정할 이유가 없고, 영문 부호화와의 구별에 그 주목적이 있으며 HFSS의 작동시 우리말 글자체의 선택을 우리말에 맞게 하도록 하는 기능을 한다.

글자체 가족 NFSS에서와 같이 우리말의 글자체 가족은 글자체의 설계 원칙에 따라 구분이 되고, 우리말의 특성에 의해 두 요소로 표현된다.

첫번째 요소는, \kscfamily로서 우리말의 상징 문자와 한글 및 한자를 구별한다. 이 매크로도 사용자의 입장에서는 쓸 일이 거의 없고, 주로 HLaTeX의 내부 처리 과정에서 사용된다.

두번째 요소는 우리말의 글자체 가족을 나타내는 \hfontfamily로서 우리말의 글자체 가족을 바꿀 때 사용자가 직면하게 되는, 실질상 우

것이다.

리말의 글자체 가족을 대표하는 요소이다.

글자체 무게 및 폭 작동 방식과 내용이 NFSS와 같다. 추가된 것은 글자체 시리즈의 접두사 S로서, 우리말의 `\bfseries`가 `\softbold`로 처리되도록 하는 데 쓰인다.

글자체 모양 NFSS와 같다.

글자체 크기 NFSS와 같다.

2.3.1 우리말 글자체의 가족 선택

KS X 1001의 부호 체계에 의한 상징 문자·한글·한자의 가족 구분은 글자의 부호값으로 정해지므로 특별한 명령·선언 매크로를 필요로하지 않는다.

단지 미래에 추가될 수도 있는 가능성 혹은 운영 체계 관리자를 위해, 설치된 HLaTeX의 특성에 따라 다음의 매크로로 상징 문자·한글·한자를 구분 설정할 수 있도록 하였다.

```
\kscfamily{상징문자}{한글}{한자}
```

이는 각각

```
\symboldefault
\hanguldefault
\hanjadefault
```

로 설정되어 있으며 그 값은 각각 `s`, `w`, `h`이다.

우리말 문서에서의 실질적인 "우리말 글자체 가족"은

```
\textmj  \mjfamily
\textgt  \gtfamily
\texttz  \tzfamily
```

로 선택되며 이들은 각각

```
\textrm  \rmfamily
\textsf  \sffamily
\texttt  \ttfamily
```

와 같은 작용을 한다. (왼쪽은 명령형, 오른쪽은 선언형)

일반적으로 우리말 글꼴 가족을 선택하는 매크로는 \hfontfamily이다. 위의 명령 또는 선언형의 매크로 명령도 이것을 이용하고 있다.

```
\hfontfamily{우 티 말 가족}
```

UHC 글꼴을 쓸 때의 우 티 말 가족은 표 2.3과 같다.

표 2.3: UHC 글꼴 우리말 글꼴가족

기본가족:					
mj	(명조*)	gt	(고딕*)	tz	(타자)
추가가족:					
gr	(그래픽)	gs	(궁서)	gh	(궁흘림**)
mg	(목각**)	pg	(필기)	yt	(옛글)
bm	(봄글씨)	pn	(펜글씨)	ph	(펜흘림)
sh	(신문)				
jmj	(자모명조)	jgt	(자모고딕)	jnv	(자모노벨)
jsr	(자모소라)				

이 우리말 가족들은 상징 문자·한글·한자의 세 가족을 대표하는 외형 가족으로서, 문서에서 우 티 말 가족을 tz로 지정하였을 경우 HLaTeX은 한글·상징 문자·한자의 순으로 각각

```
wtt, wtt, wtt
```

의 내형 가족을 쓰도록 되어 있다. 외형 가족과 내형 가족의 연관은 사용자가 \MapHangulFamily 매크로 명령으로 문서의 어디에서든지 변경할 수 있다.

```
\MapHangulFamily{외형 가족}{한글 가족,상징문자 가족,한자 가족}
```

tz의 경우 내형 가족과의 연관은 다음과 같은 선언으로 이루어진다.

```
\MapHangulFamily{tz}{wtt,wtt,wtt}
```

\textmj는 우리말의 기본 글자체로 선언되어 있고 호환성과 편의를 위해 표 2.4와 같은 추가 매크로가 정의되어 있다.

표 2.4: 추가 글꼴 가족 선택 명령과 선언

명령형	선언형
\textgr	\grfamily
\textgs	\gsfamily
\textmg	\mgfamily
\textyt	\ytfamily
\textbm	\bmfamily
\textpn	\pnfamily

HLaTeX 0.98까지 사용되던 완성형 글자체의 \smfamily와 \olfamily 및 UHC 글자체의 \shfamily는 삭제되었고 펜흘림 글자체를 선택하는 매크로 \phfamily는 만들지 않았다. 이와 같은 추가 매크로의 사용은 되도록 피하도록 하고 대신에 \hfontfamily{외형가족}을 쓰거나 2.3.5에 설명하는 우리말 매크로 명령을 쓰도록 하자.

2.3.2 우리말 글자체 시리즈(series)

문서의 글자체 시리즈는 NFSS가 그대로 적용되며 LaTeX을 쓸 때는 HLaTeX 0.96판까지 사용되었던 softbold를 위해 글자체 시리즈의 접두사 S가 추가되어 보통 두께의 글자체를(\mdseries) 세번 중첩 인쇄하는 기능을 한다. 그러므로 LaTeX 문서에서 다음의 명령은

```
  \renewcommand{\bfdefault}{Sbx}
```

혹 은

```
  \fontseries{Sbx}
```

우리말의 두께를 softbold로 처리하도록 하며 영문 글자체는 bx가 되도록 한다. hangul 패키지의 추가 선택 softbold은 위와 같은 기능을 애초값으로 정해준다.

2.3.3 우리말 글자체의 모양(shape)과 크기(size)

우리말 글자체의 모양·크기는 NFSS와 같다.

2.3.4 우리말 글자체의 애초값

문서의 애초 글자체는 NFSS와 같이 다음의 명령·선언형으로 선택된다.

```
\textnormal    \normalfont
```

HLaTeX에서 애초값은 표 2.5와 같이 설정되어 있다.

표 2.5: 우리말 글자체 애초값

	LaTeX	Lambda
\hencodingdefault	H	
\encodingdefault		UHC
\symboldefault	s	
\hanguldefault	w	
\hanjadefault	h	
\mjdefault	mj	mj
\gtdefault	gt	gt
\tzdefault	tt	tz
\hfamilydefault	\mjdefault	
\familydefault		\mjdefault

위와 같은 애초값들은 운영 체계 관리자가 hfont.cfg에서 변경 설정할 수 있고, 사용자도 문서의 서두에서 재정의할 수 있다.

2.3.5 그외 우리말 글자체의 매크로

LaTeX2.09 형식 글꼴 바꿈 선언. HLaTeX 0.96판까지 써 오던 두 글자로 구성되는 글자체 바꿈 매크로는 다음 세 개만을 남기고 모두 삭제되었다. 이들은 LaTeX2.09 식의 글자체 바꿈 매크로로서 되도록 사용하지 않는 것이 좋다.

```
\mj
\gt
\tz
```

우리말 글꼴 바꿈 선언형 매크로. 그 반면에 우리말로 구성되는 글자체 바꿈 명령은 글자체의 가족을 선택하는 선언형으로서 계속 존재한다. 표 2.6을 보라.

표 2.6: 우리말 글꼴 선택 매크로

	완성형	UHC	Lambda
\명조	\mjfamily	\mjfamily	\mjfamily
\고딕	\gtfamily	\gtfamily	\gtfamily
\타자	\tzfamily	\tzfamily	\tzfamily
\그래픽	\hfontfamily{gr}	\hfontfamily{gr}	\fontfamily{gr}
\궁서	\hfontfamily{gs}	\hfontfamily{gs}	\fontfamily{gs}
\필기	\hfontfamily{pg}	\hfontfamily{pg}	\fontfamily{pg}
\목각	\hfontfamily{mg}		
\옛글	\hfontfamily{yt}		
\봄글씨	\hfontfamily{bm}	\hfontfamily{bm}	\fontfamily{bm}
\펜글씨	\hfontfamily{pn}	\hfontfamily{pn}	\fontfamily{pn}
\펜흘림	\hfontfamily{pn}	\hfontfamily{pn}	\fontfamily{pn}
\신문		\hfontfamily{sh}	\fontfamily{sh}

한글 글자체 관련 내부 명령. 다음 명령들은 한글 글자체를 선택하는 내부 명령들이다. 시스템 관리자나 새로운 글자체 등록을 위해서 여기서 설명하지만, 일반적으로 사용자 입장에서는 크게 신경쓸 일이 없다.

\usefont는 NFSS의 매크로 명령으로서 글자체의 부호화와 가족, 시리즈 및 모양을 한번에 정할 수 있도록 한다.

```
\usefont{부호화}{가족}{시리즈}{모양}
```

이 매크로에서 첫번째 변수인 글자체 부호화가 \hfontencoding과 같을 경우에는 두번째 변수를 \hfontfamily로 취급한다. 그러므로 다음의 선언은

```
\usefont{H}{gs}{m}{n}
```

우리말 글자체중 궁서체의 보통 두께, 보통 모양을 쓰도록 한다. 이때 영문의 경우도 두께와 모양에서 우리말을 따르게 된다.

다음과 같은 명령은 NFSS에 대응하는, 우리말을 위한 선언으로서 HFSS의 작동에 필수불가결한 매크로이다. 그렇지만, 사용자의 입장에서는 보통의 경우 신경쓰지 않아도 된다.

```
\DeclareErrorHFont{부호화}{가족}{두께}{모양}{크기}
\DeclareHFontSubstitution{부호화}{가족}{두께}{모양}{크기}
```

2.4 은글꼴 및 TTF2HLaTeXFont를 이용할 때의 글자체 선택 명령

2.4.1 은글꼴 글자체 선택

위의 설정과 명령 및 선언들은 HLaTeX 0.991의 기본값으로서 UHC 글꼴을 사용하는 경우에 해당한다. hangul-k 패키지는 주로 PDF 문서 작성을 위하여 만들어졌으므로 트루타입 글꼴을 사용하게 되는데, 이 때 글꼴 선택명령은 위의 경우를 참고로 하여 작성된다.

은글꼴 통합 패키지를 설치하였을 때는 unttf.sty 패키지를 로드하여 글꼴 설정을 한다. 이 경우 사용할 수 있는 글꼴 가족은 다음 표 2.7과 같다. 이 글꼴 가족 명칭을 \hfontfamily 명령으로 선언할 수 있다. 예를 들어, 펜글씨체를 선택하려면 다음과 같이 한다: \hfontfamily{pn}.

그리고 글꼴 선택명령은 표 2.8과 같이 설정되어 있다.

표 2.7: 은글꼴 우리말 글꼴가족

기본가족:					
bt	(바탕*)	gt	(고딕)	tz	(타자)
추가가족:					
gr	(그래픽)	gs	(궁서)		
		pg	(필기)	yt	(옛글)
bm	(봄글씨)	pn	(펜글씨)	ph	(펜흘림)
sh	(신문)				
jmj	(자모명조)	jgt	(자모고딕)	jnv	(자모노벨)
jsr	(자모소라)				

2.4.2 임의의 트루타입 글꼴 설정

일반적으로 사용자가 자신의 글꼴 세트를 트루타입으로부터 만드는 방법은 TTF2HLaTeXFont 스크립트를 이용하는 것이다. 김도현 님의 이 스크립트 사용법은 매우 단순하며, 글꼴 세트의 대응만 미리 잘 정해주면 대부분의 트루타입 글꼴을 사용자가 설정하여 자신의 문서에서 활용할 수 있다.[9]

자신이 사용할 글꼴 세트의 스타일 명칭을 결정한다. 글꼴 제공사의 명칭을 쓰는 것이 일반적이지만 사용자 자신이 임의로 명명할 수도 있다. 이것을 예컨대 my라고 하자.[10]

이제, config-my라는 설정 파일을 작성한다. 그림 2.5는 이 설정 파일의 내용을 예시한 것이다.

당연히, 여기 지시된 *.ttf 글꼴 파일들은 윈도 시스템 폰트 폴더에 이미 설치되어 있거나 사용자의 작업 디렉토리에 들어 있어야 한다. 또한, 한글 트루타입 중에는 한자 영역이나 기호 영역이 없는 것도 있으므로 충분

[9] 이 스크립트를 이용하려면 Perl이 설치되어 있어야 한다. 이에 대해서는 http://faq.ktug.or.kr/mywiki/ActivePerl을 참고하라. TTF2HLaTeXFont는 http://faq.ktug.or.kr/mywiki/TTF2HLaTeXFont 페이지에서 다운로드받을 수 있다.

[10] 이 글꼴 세트 명칭의 명명이 너무나 자유로운 까닭에 다른 사람이 만든 글꼴 세트 명칭과 중복될 가능성이 너무 많다. 이 명명법에 대하여 일반적 규칙이 있는 것은 아니나, 되도록이면 글꼴 세트의 특징을 드러내면서, 중복되지 않도록 주의하는 것이 필요할 것이다.

표 2.8: 글꼴 선택 명령. 은글꼴.

기본글꼴		
바탕	돋움	타자
\mjdefault	\gtdefault	\tzdefault
(\rmfamily)	(\sffamily)	(\ttfamily)
(\mjfamily)	(\gtfamily)	(\tzfamily)
글꼴	**선언형**	**명령형**
펜흘림	\phfamily	\textph
자모소라	\jsrfamily	\textjsr
자모고딕	\jgtfamily	\textjgt
그래픽	\grfamily	\textgr
자모노벨	\jnvfamily	\textjnv
필기	\pgfamily	\textpg
옛글	\ytfamily	\textyt
궁서	\gsfamily	\textgs
신문	\shfamily	\textsh
자모명조	\jmjfamily	\textjmj
봄글씨체	\bmfamily	\textbm
펜글씨	\pnfamily	\textpn

한 주의를 기울여야 할 것이다.[11] 그리고 설정 파일에서 FOUNDRY 명칭과 FONTmj는 최소한 반드시 필요하다.

그림 2.5의 예에서 **아롱체**는 임의로 ar이라는 글꼴 명칭을 부여하였는데, 이것은 HLaTeX에는 없는 글꼴 가족 이름이다. 이런 식으로 새로운 글꼴 가족을 부여하는 것은 사용자 자신의 문서에서 사용하는 데는 지장이 없으나, 다른 문서와의 호환성에서는 문제가 될 수도 있으므로 주의해서 사용하여야 한다.

명령행에서 스크립트를 실행한다. 이제 명령행을 열고, 다음과 같이 스크

[11] TTF2HLaTeXFont 스트립트는 트루타입 글꼴의 빠진 영역을 다른 기본 글꼴로 대체해주는 기능이 있다.

```
FOUNDRY: my
# 바탕. 한컴바탕. 휴먼견출명조.
FONTmj: f=mj m=hbatang.ttf b=hmfem.ttf
# 돋움: 한컴돋움. 휴먼견출고딕.
FONTgt: f=gt m=hdotum.ttf b=hmfeg.ttf
# 타자: 아시아타자.
FONTtz: f=tz m=ktjm.ttf b=ktjb.ttf
# 아통: 아통.
FONTar: f=ar m=along.ttf
```

그림 2.5: TTF2HLaTeXFont의 config-my 예제

립트를 실행한다.[12]

```
#> perl ttf2hlatexfont.pl -c config-my
```

그 결과 만들어진 폰트 세트가 마음에 드는지를 `testlatex.tex`을 컴파일하여 확인해본다.

TEXMF tree로 설치하고 등록한다. 다음 명령으로 새로운 TEXMF tree로 설치할 수 있다. 여기서는 `C:\TEXMF-MY`에 설치하는 경우를 예로 든다.

```
#> perl ttf2hlatexfont.pl -c config-my -i C:\TEXMF-MY
```

파일 복사가 끝난 다음에 다음과 같은 추가 조치를 해주자. MiKTEX의 경우를 예로 든다.

먼저 `C:\TEXMF-MY\dvipdfm\config` 폴더를 열면 `dvipdfmx.cfg`가 있을 것이다. 이것으로 `LocalTEXMF` 아래의 texttt`dvipdfm\config` 아래 있는 `dvipdfmx.cfg`를 교체한다.[13]

[12] Lambda를 위한 `*.ofm`을 만들도록 하려면 옵션 `-o`를 추가해주어야 한다. 다만, 영문 MiKTEX 2.4 사용자는 이 옵션을 쓸 수 없다. Lambda 사용을 위해서는 MiKTEX-KTUG 2.3을 설치·사용하도록 하라. 이밖에 PDFLATEX을 위한 `*.enc`를 만들도록 하는 추가 옵션 `-p`가 더 있지만, 여기서는 문제삼지 않겠다.

[13] 물론 `LocalTEXMF` 아래 `dvipdfmx.cfg`에 `f cid-myttf.map` 행을 직접 추가하여도 된다.

그리고 C:\TEXMF-MY\ttf2pk\base 아래에 있는 ttf2pk.cfg로, 역시 LocalTEXMF 아래에 있는 파일을 교체한다.

이제 MiKTeX Options를 실행하여 Refresh Now 버튼을 눌러 Filename Database를 갱신한다.

이 이후로 문서에서 myttf.sty를 \usepackage하면 위에서 설정한 자신의 트루타입 폰트 세트를 사용할 수 있다. myttf.sty에는 위의 경우 예를 들면 \mjdefault, \gtdefault, \tzdefault 이외의 글꼴 가족에 대하여 \arfamily와 \textar이라는 선언과 명령을 정의해두고 있다.

c 시리즈. 은글꼴과 TTF2HLaTeXFont로 만들어지는 한글 트루타입 글꼴 세트는 특별한 hfontseries를 가지고 있다. 즉, NFSS의 c-시리즈가 장평 75%의 글꼴임에 비하여, 한글 트루타입에 의해 만들어지는 글꼴 세트에서는 이것이 장평 92%로 설정되어 있는 것이다. 이렇게 한 이유는 한글 문서에서 92% 장평의 문자를 식자하는 경우가 75% 장평의 문자를 사용하는 경우보다 훨씬 많다고 판단하였기 때문이다. 이 한글 c-시리즈 글꼴의 사용에 대해서는 67페이지를 참고하라.

제 3 절 우리말 설정

3.1 우리말 숫자 매크로

HLaTeX에서는 우리말 숫자를 숫자 환경에서 쓸 수 있도록 우리말 숫자 명령을 제공한다. hangul-k 패키지는 여기에 몇 가지를 추가하여 우리말 숫자 명령을 확장하였다.

다음은 HLaTeX에서 정의한 우리말 숫자 매크로 명령이다. hangul-k 패키지는 이 가운데 \Hnum에 대해서 "첫째, 둘째, 세째, 네째"로 되어 있던 것을 "첫째, 둘째, 셋째, 넷째"로 바꾸고 \hnum이 좀더 잘 작동하도록 조금 수정되어 있다.

```
\jaso(ㄱㄴㄷㄹ)  \gana(가나다라)
\ojaso(㉠㉡㉢㉣) \ogana(㉮㉯㉰㉱)
\pjaso((ㄱ)(ㄴ)(ㄷ)(ㄹ)) \pgana((가)(나)(다)(라))
\onum(①②③④)  \pnum((1)(2)(3)(4))
\oeng(ⓐⓑⓒⓓ)  \peng((a)(b)(c)(d))
\hnum(하나둘셋)  \Hnum(첫째둘째)
```

다음은 hangul-k 패키지에서 추가한 우리말 숫자 매크로 명령이다. LaTeX에서 \roman이나 \Roman으로 정의되는 것과 유사한 결과이지만, 한글 글꼴에 있는 상징문자를 가져다 쓰게 함으로써 자동조사와 같은 추가적인 기능을 이용할 수 있도록 하려고 정의한 것이다.

```
\hroman(ⅰ ⅱ ⅲ ⅳ)  \hRoman(Ⅰ Ⅱ Ⅲ Ⅳ)
```

우리말 숫자 매크로는 enumerate와 같은 환경에서 쓸 수 있다.[14] 다음 보기는 \labelenumi를 재정의하여 우리말 숫자를 사용한 예이다.

```
\renewcommand\labelenumi{%
    \gana{enumi})}
\renewcommand\labelenumii{%
    \Hnum{enumii},}
\begin{enumerate}\돋움
\item 한글
\begin{enumerate}
\item 현대어 한글(완성형).
\item 현대어 한글(UHC).
\item 모든 한글(유니코드).
\end{enumerate}
\end{enumerate}
```

가) 한글

첫째, 현대어 한글(완성형).

둘째, 현대어 한글(UHC).

셋째, 모든 한글(유니코드).

[14] 우리말 숫자 매크로를 장절명령의 장절 번호(section number)로 사용하는 데는 조금 어려움이 따른다. 처음부터 장절 번호로 사용할 수 있도록 설계되지는 않았기 때문이다. 이것을 가능하게 하는 부가 패키지로 ganasection이라는 스타일이 있다. http://faq.ktug.or.kr/mywiki/HLaTeXStylePackages 페이지를 참고하라.

3.2 우리말 색인

색인(찾아보기)를 만들기 위해서는 makeidx 패키지와 makeindex 외부 프로그램을 사용한다. 예컨대 문서의 Preamble에,

```
\usepackage{makeidx}
\makeindex
```

를 선언하면 latex이 실행되면서 \index 명령에 의해 지시된 단어와 어구를 모아서 .idx 확장명을 갖는 보조파일을 만든다. 이 보조 파일에 대하여 makeindex를 실행하면 theindex 환경 안에 색인항목(엔트리)들이 정렬된 .ind라는 파일이 만들어진다. 이 파일의 내용을 문서에서 불러오려면 \printindex 명령을 써주면 된다.

우리말 색인 작성은 EUC-KR 부호계로 만들어진 .idx 파일이 만들어져야 한다. 이를 위해서 latex 실행시에 8비트 문자 출력을 설정해둘 필요가 있다.[15]

이 .idx 파일의 우리말화된 정렬(sort)을 위해서 hind.ist가 준비되어 있다. makeindex 명령을 실행할 때, 색인 스타일 파일을 다음과 같이 불러준다.

```
#> makeindex -s hind.ist <파일이름>
```

makeindex 프로그램 자체는 글자를 정렬할 때 우리말을 제대로 인식하지 못한다. 그저 단순히 영문 ASCII 부호 체계의 순서에 의해 글자를 정렬할 뿐이다. 우리말은 ASCII 부호값이 161에서 시작하므로 영문의 색인이 영문자별로 구분되어 맨앞에 나타나고 우리말은 우리말 부호 체계의 값에 따라 구분되어 영문에 이어 자소의 구분없이 나타나게 된다.

[15] 8비트 문자 출력을 위해서 문서의 첫 줄에 %& --translate-file=cp8bit.tcx를 선언하거나, latex의 실행 옵션으로 --translate-file=cp8bit.tcx를 주면 된다. 첫 줄이 %&로 시작할 때 이를 반영하도록 하는 것은 TEX System의 설정(configuration) 사항이다. 다행히 MiKTEX에서는 8비트 문자 출력이 기본값으로 되어 있으므로 별도의 설정 없이 그냥 latex을 실행하여도 8비트 문자 출력을 얻을 수 있다.

hind.ist는 우리말 상징 기호와 한자를 각각 하나의 집단으로 처리하고 한글은 자소에 의해 나뉘도록 해준다. 그리고 영문은 알파벳으로 구분되어 목록이 작성되고 한글은 자소에 의해 목록이 작성된다.

영문 영역에 나타날 우리말을 우리말의 자소에 의해 구분되는 한글 영역으로 옮기기 위해서는 문서에서 \index 명령을 그림 2.6과 같이 사용하는 방법이 있다.

\HLaTeX\index{한글@\HLaTeX}	←	\HLaTeX은 '한글LaTeX'으로 인쇄된다. 그러나 문자열이 '\'로 시작하기 때문에 영문자 부호 '\'에 '한글LaTeX'이 위치하게 된 것이다. 이 때는 '@' 글자의 앞에 한글이라 적음으로써 그 위치가 '한글'이 되도록 할 수 있다.

그림 2.6: \index 명령의 사용례

색인 목록의 머릿글자 틀은 해당하는 자소가【와 】사이에 두꺼운 글씨체로 쓰인다. 이 문서의 찾아보기를 참고하라.

머릿글자는 다음과 같은 정의에 의해 식자되었다.

```
\newcommand\hindexhead[1]{【{\bfseries#1}】\hfil}
```

이 \hindexhead를 재정의(\renewcommand)하면 색인 목록의 머릿글자 모양을 바꿀 수 있다.

색인 작성과 같은 방법으로 작동되는 어휘집(glossary)은 원래 LaTeX에서 제공되지 않는다.

그러므로 어휘집을 작성할 때는 사용자가 직접 어휘 환경을 정의해야 한다. HLaTeX에서는 우리말 문서 작성의 편의를 위해 나름대로 적당한 어휘 환경을 제공한다. 색인과 어휘집을 만드는 일반적인 방법을 비교한 표 2.9를 보라.

표 2.9: 색인과 어휘집

	색인	어휘집
Preamble	\usepackage{makeidx}	같음
보조파일 명령	\makeindex	\makeglossary
항목 추가	\index	\glossary
환경 포함하기	\printindex	\printglossary
외부 명령	makeindex	makeindex[a]

[a]옵션을 다음과 같이 한다. `-o foo.gls foo.glo`

색인 형식의 어휘집으로 충분하지 않아서 사용자가 직접 어휘 환경을 정의하고자 할 때는 theglossary 환경을 재정의(\renewenvironment)해야 한다는 점에 유의하라.[16]

색인 작성에 쓰이는 \printindex와 같은 역할을 하는 \printglossary는 확장명이 .gls인 파일을 읽도록 되어 있다.

어휘집을 만들 때는 makeindex를 다음과 같이 부른다.

```
#> makeindex -s hglo.ist -o <파일이름>.gls <파일이름>.glo
```

hglo.ist는 어휘환경의 스타일 파일이다.

어휘 환경에서는 어휘 환경의 특성을 사용자가 알맞게 조정할 수 있도록 하고자 색인 환경에서와는 달리 \item 대신에 \gloitem을 쓰도록 되어 있다. 기본값은 \item과 같다.

[16]어휘집을 만드는 데 많이 쓰이는 방법으로는, glosstex이 있으나 안타깝게도 한글 처리가 완전하지 못하다. 그밖에 acronym, nomencl 패키지들은 설명붙은 어휘집을 만드는 데 도움이 될 것이다. 이 패키지들은 모두 사용방식이 조금씩 다르므로 반드시 패키지 문서를 잘 읽어보아야 할 것이다.

3.3 우리말 이름

HLaTeX 0.99 버전부터는 LaTeX에서 사용되는 각종 이름을 변경하는 \ksnamedef 명령을 제공한다. 예를 들어, LaTeX에서 사용되는 '참고 문헌'의 이름 'Reference'는 \refname에 지정되어 있다. 이를 우리말화하기 위하여 다음과 같이 쓴다.

```
\ksnamedef{refname}{참 고~문 헌}
```

표 3.3에 HLaTeX에서 한글화한 "LaTeX 이름의 한글 및 한자화"가 나열되어 있다. 이 한글화는 위의 예와 같이 \ksnamedef로 재정의하면 원하는 대로 변경할 수 있다.

HLaTeX의 기본 한글 명칭 가운데 "목차"보다는 "차례", "참고 서적" 보다는 "참고 문헌"이 더 많이 쓰이지 않느냐는 논의가 있었다. hangul-k 패키지는 HLaTeX의 기본값을 변경하지는 않았고, 사용자가 \ksnamedef으로 이 값을 선택하여 쓰도록 하였다.

3.4 우리말 장절명령

ksTHE는 LaTeX에 정의되어 있지 않고, HLaTeX에서 편장절의 번호 숫자를 짜는데 사용되도록 도입된 이름이다. 이는 장절명령을 한글·한자화함으로써 발생하는 문제점들을 해결하기 위해 사용된다.

예를 들어 \part의 일련 번호는 LaTeX 방식으로 그냥 쓰면 "편 I"과 같이 나오기 때문에, HLaTeX에서는 "제 I 편"과 같은 식으로 짜여지도록 장절 명령들을 재정의하고 있다.

HLaTeX에서는 \part, \chapter, \appendix, \section까지를 한글화하고 있으며, 각각 다음과 같이 설정되어 있다.

이를 사용자의 편의에 따라 재설정하기 위해서는 다음 명령을 사용한다.

```
\kscntformat{단위 이름}{앞}{뒤}
```

예를 들어 \chapter를 "첫째 마당"과 같은 식으로 짜기 위해서는 Preamble에서 다음과 같은 설정을 하면 된다.

표 2.10: LaTeX 이름의 한글 및 한자화

LaTeX명령	HLaTeX(한글)	HLaTeX(漢字)
contentsname	목~록	目~錄
abstractname	요~약	要~約
listoffigurename	그림~목록	그림~目錄
listoftablename	표~목록	表~目錄
bibname	저서~목록	著書~目錄
refname	참고~서적	參考~書籍
indexname	찾아보기	索~引
partname	편	篇
chaptername	장	章
sectionname	절	節
appendixname	부록	附錄
ksTHE	제	第
today	1994년 3월 6일	1994年 3月 6日
pagename	쪽	쪽
tablename	표	表
figurename	그림	그림
seename	\을 참조	\을 參考
ccname	사본	寫本
enclname	동봉물	同封物
headtoname	받는이	受信人
glossaryname	용어~풀이	語~彙
colorlayer	환등판~색깔	幻燈版~色相

```
\renewcommand{\thechapter}{\Hnum{chapter}}
\ksnamedef{chaptername}{마당}
\kscntformat{chapter}{}{~\chaptername}
```

hangul-k 패키지는 hangul의 장절명령 설정을 그대로 계승하고 있다. 그런데 장절명령을 변경하거나 수정하는 패키지들, 예컨대 sectsty와 같은 스

표 2.11: 장절명령의 한글화

단원 종류	숫자 앞	숫자	숫자 뒤
\part	\ksTHE~	\thepart	~\partname
\chapter	\ksTHE~	\thechapter	~\chaptername
\appendix	\appendixname~	\thesection \thechapter	article book/report
\section	\ksTHE~	\thesection	~\sectionname

타일을 쓰면 hangul의 설정은 더이상 효력이 없어서 “장 1”과 같은 결과가 나타나게 된다. 사용자가 장절명령의 스타일을 직접 바꾸려면 hangul의 설정을 재정의하여야 하는데, 이것이 번거로운 일이었다. hsectsty는 이러한 번거로움을 조금 줄여보려는 노력의 소산이다. 72페이지를 보라.

제 4 절 자동 조사 처리

hangul-k 패키지가 hangul 패키지와 크게 달라진 점이 자동조사이다. hangul-k 패키지의 자동조사 기능은 전적으로 PDF를 제작하기 위한 목적에서 비롯되었고, 따라서 hyperref 패키지에 크게 의존하고 있다.

이 단원에서는 hangul-k 패키지의 자동조사 기능을 개관하고 사용법을 요약한다.

4.1 HLaTeX의 자동조사 구현

자동조사 처리는 hLaTeXp에서부터 구현되어 있었다. LaTeX을 이용한 문서 작성 상황에서 상호참조와 인용을 처리하기 위해서 자동조사는 필수적인 도구였다고 할 수 있다.

예컨대, ‘문제 \ref{삼각함수}를 보면\ldots’와 같이 글을 쓰는 경우, \ref{삼각함수}라는 참조명령이 만들어낼 숫자가 어떤 것이 될지를 미리 알지 못하기 때문에 ‘을’을 써야할지 ‘를’을 써야할지 문서 작성시에

어려움을 겪게 된다([차재춘1998], p. 6).

hLaTeXp와 HLaTeX은 자동조사 명령이 불릴 인용표지(*reference key* 또는 *cite key*)에 미리 "자동조사 생성 명령"을 붙여서 보조파일(.aux)에 기록해두었다가 이것이 참조되는 시점에서 생성된 문자(열)의 형태를 "ㄹ 이외의 종성(\@jong), 중성(\@jung), 받침 ㄹ(\ri@el)"이라는 표지로 구분하여 자동조사를 붙여주는 방식으로 작동하였다.(hLaTeXp의 경우에는 이 표지명령의 명칭이 \k@fson 등으로 달랐을 뿐이고 구현 방법은 거의 유사하였다.)

이러한 자동조사 명령은 몇 가지 특별한 경우를 제외하고는 매우 잘 작동하였으며 HLaTeX의 장점 가운데 하나였다. 그러나 문제는 다음과 같은 경우에 발생하였다.

영문 스타일 패키지와의 충돌. 일부 영문 스타일 패키지는 보조 파일(.aux)을 읽어서 원하는 동작을 하는 경우가 있다. 예를 들면 MSC 패키지나, AcroTeX, cite, overcite, prosper, verse 등의 패키지들이 그러하였다. 이런 패키지를 사용하는 경우 .aux에 포함된 자동조사 처리를 위한 코드들이 패키지의 내부 코드와 충돌(불일치)하거나, 원하지 않는 동작을 하는 결과를 가져오는 경우가 많았다. hangul 패키지를 쓰면 오류가 발생하지만 hfont로는 문제가 없다면 대부분이 이러한 경우라고 볼 수 있다.

hyperref의 하이퍼링크 기능과의 충돌. 결정적으로 문제가 되었던 것은 PDF를 만들면서 하이퍼링크를 hyperref 패키지로 구현하려 할 때였다. hyperref 패키지는 hangul과 함께 쓸 수 없었는데, 그 이유가 hyperref 자체가 보조 파일 .aux를 매우 크게 변형시키고 있기 때문이었다. 그 가운데서도 문제가 되었던 것은 숫자의 자동조사를 결정하기 위한 \ks@num이라는 함수였는데, 이것이 .aux에 그대로 포함되면서 자동조사가 동작하지 않는 것은 물론이고, 하이퍼링크 이외의 다른 여러 동작까지도 영향을 미쳐 오류를 발생시키곤 하였다. [nojosa] 옵션을 추가하는 것만으로는 이 오류를 없애기가 매우 어려웠다.

hangul-nojosa 사용. 그래서 일부에서는 차라리 자동조사를 포기하더라도 PDF 하이퍼링크를 구현하려는 생각으로 hangul에서 자동조사 관련 코드를 제거하여 hangul-nojosa라는 스타일을 만들게 된다. 자동조사 기능이 LaTeX 한글화의 핵심 가운데 하나임을 생각할 때, 이런 접근방법은 아무리 궁여지책이었다고 하나 바람직스럽지 못한 것이었음이 명백하다. 자동조사 없는 한글LaTeX이란 앙코 없는 찐빵이 아니겠는가.

hangul-k 패키지. 2004년 5월에 김도현 님에 의하여 아이디어가 제시되고 마침내 수일 간에 걸쳐서 선을 보인 hangul-k 패키지는 hangul-nojosa에 자동조사 기능을 구현해넣은 것이다. hangul-k 패키지의 자동조사는 hangul 패키지의 방식과는 기원이나 발상이 다르고 따라서 독자적인 해결책을 이루고 있다.

4.2 hangul-k 패키지의 자동조사 처리

4.2.1 자동조사명령

hangul에서 제공하는 자동조사 명령은 다음과 같은 것이 있었다.

\은, \는, \이, \가, \을, \를, \와, \과, \로, \으로

이 열 가지 자동조사 명령에서 \은, \는 등이 따로 있는 것은 입력의 편의를 위한 것일 뿐이고, \은과 \는, \이와 \가는 그 기능이 동일하다. 다른 것도 마찬가지다.

hangul-k 패키지는 여기에 \라, \이라 조사 명령을 추가하였다. 서술격 조사 어간 이는 그 앞에 오는 말이 모음으로 끝날 때 줄어질 수 있기 때문에 비단 이라뿐 아니라 이고, 이니, ... 등에서도 이따금 이가 줄어질 수 있겠지만, 라, 이라의 경우와 같이 일반적이 아니라고 판단하여 이 경우는 라, 이라만을 자동조사의 범위에 포함시킨 것이다.

우리말의 조사 규칙에 대해서는 78페이지로 가서 표 4.1을 참고하라.

hangul-k 패키지에 새로 추가된 우리말 숫자 명령 \hroman과 \hRoman은 자동조사의 처리를 위해서 필요하였다. 영문의 \roman이나 \Roman으로 얻어지는 자동조사를 붙여야 할 문자열은 영문 알파벳 i 또는 ii이었기 때문에 이것을 "아이"로 읽는 한 자동조사의 작동을 보증하지 못한다. 한글 상징기호로 이 문자들을 식자하면 거기에 자동조사를 붙일 수 있게 되는 것이다.

```
\renewcommand\theenumi{%
  \hroman{enumi}}
\begin{enumerate}
\item\label{test:first}
  첫번째 아이템.
\item\label{test:second}
  두번째 아이템.
\end{enumerate}

위의 \ref{test:first}\를 보면,
```

ⅰ. 첫번째 아이템.

ⅱ. 두번째 아이템.

위의 ⅰ을 보면,

\hroman과 \hRoman에 자동조사 명령을 쓸 경우에는 josa.tab을 교체하여야 한다. HLaTeX의 기본 josa.tab에는 이 상징문자들의 조사 값이 지정되어 있지 않기 때문이다.

4.2.2 참조형 자동조사

참조와 인용은 대부분 hyperref 패키지에 의해서 하이퍼링크가 자동으로 생성된다. hangul-k 패키지는 우선 hyperref이 하이퍼링크를 만들기를 기다려서 거기서 생성되는 상호참조의 문자열을 분석하고, 그 가운데 마지막 몇 글자를 얻어서 거기에 따라 조사를 붙여주는 방법으로 구현된다. 이 과정은 자동으로 처리되기 때문에 사용자가 신경쓸 부분은 별로 없다. 다만 *cite_key*나 *ref_key*를 영문으로 사용해야 한다는 점만 주의하면 될 것이다.[17]

[17] hangul에서는 한글 참조 이름표를 사용할 수 있었다. hyperref은 한글 참조 이름표에 대해서 오류를 발생시키기 때문에, Lambda가 아니라 LaTeX을 쓰는 한 영문 이름표(label)는 피할 수 없다.

먼저 참조를 위하여 \label 명령으로 참조 키를 지정한다. 그런 다음 이것을 \ref나 \pageref 명령으로 불러온다.

```
다음 표~\ref{tab:josa}\를
참고하라.
```

다음 표 4.1을 참고하라.

4.2.3 인용형 자동조사

hLaTeXp나 HLaTeX이 모두 \cite에 의하여 이루어지는 인용의 자동조사 처리가 완전하지 못하였다.

그리 되었던 까닭은 \cite 또는 \citep, \citet 등이 만들어내는 문자열이 고정되어 있지 않고 인용 처리 방식의 변화에 따라 유동적이었기 때문이다. 말하자면 thebibliography 환경 안의 \bibitem에 붙이는 이름에는 아무런 제약이 없었고, natbib이나 그밖의 다른 bibliography 관련 패키지에 따라 얻어지는 참조형의 문자열이 자유롭고 다양하기 때문이었던 것이다.

hangul-k 패키지가 채택한 자동조사 처리 방식에 따르면 우선 \cite 명령은 hyperref에 의하여 특정한 문자열 또는 숫자로 식자된다. 이 문자열을 취하여 거기에 자동조사를 붙이는데, 문자열이 숫자와 한글인 경우라면 큰 문제 없이 자동조사 처리를 할 수 있겠지만, 영문인 경우에는 철자와 발음이 일치하지 않는 경우가 있어서 곤란을 겪을 수 있다. 이럴 경우에 대비한 몇 가지 사례는 hangul-k 패키지에 구현되어 있으나, 완전한 동작을 보증하지는 못한다.

숫자의 경우 문제가 없다는 점을 이용하면 저자-연도 방식의 경우 마지막 문자가 숫자이기만 하면 자동조사의 처리가 잘 이루어질 것이다. 그러므로, 과학 학술논문에 흔히 쓰이는 가장 단순한 형식, 예컨대 [1]과 같은 숫자 인용 형태에서는 자동조사가 잘 작동한다.[18]

[18] 숫자의 경우, 마지막이 0으로 끝나는 숫자 가운데, 1조와 10조, 100조, 1000조, 1해 등의 숫자에 대해서는 자동조사가 잘못 붙을 수 있다. 그러나 참조 문자열이 1조(1,000,000,000,000)가 되는 것은 극도로 예외적인 경우가 될 것으로 생각한다.

```
\cite{hltxman}\와
\cite{hlguide}\은
한글\LaTeX{} 사용설명서들이다.
```

[차재춘1998]과 [은광희2000]은 한글LaTeX 사용설명서들이다.

4.2.4 비참조형 자동조사

비참조형 자동조사란, 임의의 문자열에 붙이는 자동조사를 말하는 것이다. 우리는 이것을 hyperref에 의하여 상호참조 하이퍼링크가 만들어지지 않는 자동조사라는 뜻으로 쓰겠다.

임의의 사용자 카운터. 예컨대, 사용자가 문서에서 임의의 카운터를 정의해서 쓰고 있다고 하자. 이 카운터를 \label을 붙여서 호출하지 않고 카운터 뒤에 직접 자동조사를 붙여야 하는 경우를 생각해보자.

```
\newcounter{usrcnt}
\setcounter{usrcnt}{1}
\theusrcnt\가
```

위와 같은 상황에서 \theusrcnt는 '1'로 치환되지만 \label을 \ref로 부른 것이 아니기 때문에 하이퍼링크는 만들어지지 않는다. 따라서 직접 자동조사가 작동하지 않을 것이다.

이와 같은 비참조 사용자 숫자에 대하여 자동조사를 붙이기 위해서 hangulk 패키지는 몇 가지 수정된 한글 숫자 명령을 제공한다. 표 2.12는 그 명령을 예시하고 있다.

예컨대, 다음과 같은 보기를 보자. 보통 \thetmpcnt라고 써야 할 곳에 \HArabic을 지시하였다.

```
\newcounter{tmpcnt}
\setcounter{tmpcnt}{1}
\HArabic{tmpcnt}\가
\stepcounter{tmpcnt}
\OJaso{tmpcnt}\를
```

1이 ㉡을

표 2.12: 카운터 명령

한글카운터명령	비참조카운터명령	비고
\jaso	\Jaso	ㄴ
\pjaso	\PJaso	㈂
\ojaso	\OJaso	㉡
\gana	\Gana	나
\pgana	\PGana	㈏
\ogana	\OGana	㉯
\onum	\ONum	②
\pnum	\PNum	(2)
\hnum	\HNUM	둘
\Hnum[a]		둘째
\peng	\PEng	(b)
\oeng	\OEng	ⓑ
\hroman[b]	\Hroman	ii
\hRoman	\HROMAN	II
	\HArabic	2
\alph[c]	\Halph	b
\Alph	\HAlph	B
\roman		ii
\Roman		II

[a] \Hnum에는 자동조사가 붙지 않음.
[b] \hroman과 \hRoman은 hangul-k 패키지에 추가된 명령임
[c] 이 아래 네 개의 명령은 LaTeX 명령임

임의의 문자열 자동조사. 사용자가 임의의 문자열에 미리 자동조사를 위한 \make@josa를 붙여둘 수도 있다. 이를 위하여 \xjs 또는 \fjs를 쓴다. \fjs는 \jong, \jung, \rieul 가운데 하나를 인수로 가지며, 각각의 인수가 의미하는 바와 같이 마지막 문자의 종성이 ㄹ일 때 \rieul, 중성일 때 \jung과 같이 지정해두는 것이다. 반면 \xjs는 인수로 지정된 문자열 자체를 반환하면서 마지막 글자에 대해 자동조사 처리를 하도록 한다.

예컨대, \PageName과 \AltPageName은 다음과 같이 hangul-k 패키지에 미리 정의되어 있다.

```
\def\PageName{\xjs{페이지}}
\def\AltPageName{\xjs{쪽}}
```

여기서 \xjs 명령은 인수에 대하여 자동조사 처리를 위한 표지를 붙이라는 것이다. 그러므로 다음과 같은 결과를 얻게 됨을 알 수 있다.

```
20\PageName\은
20\AltPageName\은
```

20페이지는 20쪽은

다음과 같은 예에서는 \fjs의 사용법을 확인할 수 있다.

```
\renewcommand\hangulk{%
  \texttt{hangul-k}
  패키지\fjs{\jung}}
\hangulk\과 \\
\renewcommand\hangulk{%
  \textit{hangul-k}
  스타일\fjs{\rieul}}
\hangulk\과
```

hangul-k 패키지와
hangul-k 스타일과

장절명령과 \bibitem 안에서의 자동조사 명령. 장절명령의 인자로 \xjs가 붙은 명령은 최신 버전의 hangul-k 패키지가 잘 처리한다. 다만 한 가지 주의할 것은, 장절명령 안에서 자동조사 명령을 쓰는 경우이다. 예컨대,

```
\section{\hangulk\를 사용하기}
```

과 같은 지정은 에러를 내면서 실행이 멈춘다. 이럴 경우에는 자동조사 명령을 \protect 해주고, 만약 한글 책갈피를 만드는 경우라면 \texorpdfstring 명령으로 책갈피용 문자열을 별도로 지정해주는 것이 좋다. 즉, 다음과 같이 하는 것을 권장한다.

```
\section{\hangulk\texorpdfstring{\protect\틀}{} 사용하기}
```

이 방식은 좀 복잡하지만 아무튼지 에러없이 처리할 수 있다는 점에서 여기서 지적해둔다.

\bibitem의 경우에는 \fjs보다 \xjs 명령을 쓰기를 권장한다.

```
\bibitem[\xjs{박지원}]{parkjw} 박지원, \emph{소단적치인}.
```

이 글에서 다루어진 일반적 텍스트에 자동조사를 붙이는 기능은 Lambda 기반의 DHHangul에서는 별다른 고민없이 가능한 것을 확인할 수 있을 것이다. 궁극적으로는 Omega와 Lambda로 한글 LaTeX 환경이 이행하면 더 좋은 해결책이 틀림없이 나올 것으로 믿어 의심치 않는다. 다음 장에서 Lambda와 관련된 몇 가지를 알아보자.

제 3 장

Lambda와 hangul-k

제 1 절 왜 Lambda인가?

Lambda(Λ)란 Omega 기반의 LaTeX을 말한다. Omega와 Lambda에 대해서는 Omega System WWW 페이지를 참고하라. KTUGFaq Omega 페이지도 함께 참고할 수 있다. e-TeX에서 파생된 Omega인 e-Omega, 즉, Aleph 기반의 LaTeX은 Lamed[1]라고 하는데, 여기서는 구별하지 않고 Lambda로 통칭하겠다.

HLaTeX의 저자인 은광희 님은 [은광희2000]에 다음과 같이 적고 있다.

> 그러나 근본적인 문제는, TeX자체가 7비트 부호 체계를 바탕으로 만들어져 있다는 데에서 발생합니다. TeX 3판의 출현으로 유럽 언어와 같은 8비트 부호 체계의 처리는 해결되었지만 우리말과 같은 16비트 부호를 처리하는 데에는 아직 많은 문제점을 안고 있습니다. 이러한 문제의 근본적인 해결책은 16비트 부호체계를 바탕으로 형성

[1] 영문자 L에 해당하는 히브리문자(ל) 이름. Aleph가 히브리 문자 첫 글자(א) 이름인 데서 나왔다. "레임드"라고 읽어서는 안되고 "라메드"라고 읽어야 한다.

> 된 TEX의 구현으로 이루어질 수 있습니다. 세계적인 추세도 각 나라의 모든 언어를 부호화할 수 있는 통일된 부호화 방식이 필요하게 되었고, 국제 표준화 단체는 UCS를 발표하여 32비트 코드체계인 국제 부호화 문자 세트를 만들었습니다. 이에 따라 TEX에서 이 문자 세트를 사용할 수 있도록 하기 위해 8비트 코드 체계를 확장하여 16비트 코드 체계에 바탕을 둔 Omega라는 이름의 새로운 TEX이 나오게 되었습니다. 그러므로 Omega는 16비트 TEX의 구현으로서, Unicode를 처리할 수 있는 TEX이라고 간단히 표현할 수 있습니다.
>
> ...(중략)...
>
> 16비트 부호를 처리하는 TEX은 우리말과 밀접한 관계가 있습니다. 그러므로 Omega와 LATEX의 관계는 밀접할 수밖에 없습니다. Omega를 사용하여 우리말 부호를 처리할 때에, 지금까지 LATEX에 커다란 문제로 남아있는, 기존의 TEX을 바탕으로한 2×8비트식 부호 처리는 간단히 해결됩니다. Omega 1.5판으로 우리말 KS X 1001 문서를 처리할 때에 그 처리 속도는 TEX으로 영문 문서를 처리하는 속도와 차이점이 거의 없습니다. Omega는 이제 거의 완숙한 단계에 도달하고 있고 앞으로는 TEX을 완전히 대체하고 본격적으로 사용되는 방향으로 전개될 것으로 희망하고 있습니다.

순전히 기술적인 관점에서만 말한다면 Omega의 사용에 큰 어려움은 없는 단계에 와 있는 듯하다. 그러나 이 글을 쓰는 현재까지 Omega·Lambda가 일반 사용자의 문서 작성에서 범용으로 쓰일 정도까지 와 있지는 못한 것이 현실이다.

원인은 여러 가지가 있겠지만, 컴퓨팅 환경 자체의 문제가 가장 크지 않는가 한다. 최근의 대부분의 `Linux` 시스템은 UTF-8의 유니코드 환경으로 이행하였다. 그러나 가장 많은 사용자를 가지고 있는 윈도는 시스템 내부적으로만 유니코드가 작동하지 사용자의 말단 인터페이스에서는 적어도 한글 유니코드가 일반적으로 쓰이고 있지는 않다. 만약 최종 사용자의 텍스트 입출력이 UTF-8이든 UTF-16이든 유니코드로 이루어지도록 윈도 시스템이 작동하였다면 Omega의 도입은 훨씬 빨리 이루어졌을 수도 있었으리라 생각한다.

두번째는 Omega 시스템 자체의 문제인데, 이론적이 아니라 실용적인 관점에서 Lambda로는 표현할 수 있는 언어가 한정되어 있었고, 따라서 흔히

"Omega를 쓰면 모든 언어를 다 표현할 수 있다고 하더니 왜 안 되는 거죠?"와 같은 질문이 나오게 되었다. Omega는 TEX 시스템일 뿐이고 실제로 다국어 문서를 작성하려면 글꼴을 비롯하여 거기에 맞는 여러 가지 설정들을 해주어야 하는 번거로움이 상당하였던 것이다. Omega·Lambda가 직접 지원을 제공하는 언어는 현재 몇 되지 않는다.[2]

그럼에도 불구하고, 우리는 종국적으로 한글 및 CJK를 포함하는 다국어 문서는 Omega 환경으로 이행해야 하고, 언젠가는 그리 될 것이라는 희망을 가지고 있다.

제 2 절 Lambda에서 한글 구현: 개관

Omega 또는 Lambda를 사용한다면, 적절한 폰트 및 관련 설정을 해주었을 때 한글을 문서에서 표현하는 것은 크게 어려운 일이 아니다.[3] 적어도 입력 문서가 유니코드(UTF-8 등)일 경우에 그러하다. Vim이나 Yudit, SCUnipad와 같은 유니코드 편집기를 윈도에서도 이용할 수 있고 대부분의 편집기가 유니코드를 지원하므로 유니코드 입출력 자체는 큰 문제가 아닌 단계에 왔다고 보아도 좋을 것이다.[4]

2.1 한글 코드 및 입력 인코딩

Lambda 사용을 고려할 때, 한글 코드와 입력 인코딩은 중요한 문제가 된다.

---

[2] 영어, 유럽어, 아랍어, 그리스어 정도가 시스템이 직접 지원하는 언어이고, Omegaj와 같이 각국 언어를 개별적으로 Omega에서 구현하기 위한 local한 노력들은 지속적으로 이루어지고 있다.

[3] TEX으로는 이 자체가 지난한 일이었다. LATEX에서 한글을 "찍을" 수 있게 된 것만도 기적과도 같은 일이었다는 느낌을 지금도 가지고 있는 사람이 없지 않을 것이다.

[4] 다만, 중세 한글과 "첫가끝" 한글은 다른 문제이다. 현재 "첫가끝" 코드를 입력하고 편집할 수 있는 편집기는 Mozilla와 Yudit 정도라고 알고 있다. 이 둘 모두 신정식 님의 기여로 이루어진 일이다. 한글 중세 문자의 처리에 있어서 코드 문제는 아직도 미해결의 심각한 문제이다.

CP949. 현대 한글 음절. 윈도 사용자는 부지불식간에 CP949 한글을 사용하지 않을 수 없는 조건이므로, 선의의 윈도 사용자를 고려한다면 CP949 인코딩의 한글 문서를 처리"해 주는" 것은 하나의 서비스가 되지 않을까 한다. 그리 함으로써 "똠"이나 "샲"과 같은 글자가 식자되지 않는다는 불만을 줄여줄 수 있을 것이기 때문이다.

KTUG에서 2002년에 발전한 HLaTeX은 마침내 Lambda로 CP949 한글을 처리할 수 있는 능력을 갖추게 되었다.[5]

UTF-8. 현대 한글 음절. 유니코드로 한글 문자를 표현하는 방법은 두 가지가 있다. 그 하나는 이른바 "유니코드 한글"을 사용하는 것이고, 다른 하나는 "첫가끝"이라 불리는 방식이다. "유니코드 한글"이란 현대 한글 11172글자를 차례로 나열한 것으로, Unicode 2.0에서 [U+AC00]부터 가나다순으로 배치되어 있는 것을 말한다.[6]

유니코드 한글로 UTF-8의 입력 인코딩을 사용하면 현대 한글 모든 문자를 표현할 수 있다.

"첫가끝"이란, ISO/IEC-10646:1993의 24장에서 설명되고 있는 "Hangul Syllable Composition Method"를 가리킨다. 이 방법은 "Hangul Jamo" ([U+1100]–[U+11FF])에 정의된 자모 문자[7]를 첫소리, 가운뎃소리, 끝소리 순으로 결합하여 한글을 표기하는 것이다.

"첫가끝" 방식을 사용하면 현대 한글은 물론이고 중세 한글까지 완벽하게 표현할 수 있고, 한글 창제 원리에도 적합하다. "첫가끝" 방식으로 조합

[5] 여기에 기여하신 분들은 조진환, 신정식 님이었고, 은광희 님이 HLaTeX을 업그레이드하였다. 이 "진화된" HLaTeX은 CTAN 공식 배포판에 비교하여 몇 가지 파일을 더 가지고 있으며, 그 가운데 CP949 처리 부분은 MiKTeX-KTUG에 포함되어 배포되고 있다. MiKTeX-KTUG을 사용하지 않으시는 분들은 KTUGFaq Omega 페이지를 참고하여 자신의 시스템에 이 파일들을 별도로 설치하여야 한다.

[6] http://www.unicode.org/charts/PDF/UAC00.pdf

[7] http://www.unicode.org/charts/PDF/U1100.pdf 여기에는 신정식 님이 지적하신 바와 같이 U-YEO 겹모음이 빠져 있다. "바뀌었다"를 줄여쓰는 "꿨" 자의 중성 모음군에 해당하는 자소가 없는 것이다. 국제 표준을 제정할 때 좀더 융의주도했더라면 좋았을 것이라는 아쉬움이 남는 대목이다.

할 수 있는 글자의 수는 무려 160만여자에 이른다.

UTF-8. 중세 한글 문자. 1933년 한글 맞춤법 통일안에 의하여 사용하지 않기로 한 아래아를 포함하여 현대 한글에서 사용되지 않는 중세 문자를 표기하는 방법은 "첫가끝" 방식이 표준에도 맞고 올바르다. 이 점에는 이론의 여지가 없어 보인다.

문제는, 현재 널리 쓰이고 있는 중세 문헌의 전자화 방식이 이 표준을 따르지 않고 있다는 것이다. 워드 프로세서 중에서 중세 문헌을 편집하는 데 널리 쓰이는 것은 흔글과 MS Word 정도가 있는데, 이 두 가지 프로그램은 각각 한컴바탕/한컴돋움 폰트와 새바탕/새굴림이라는 폰트에 의존하고 있다. 즉, 이 폰트의 사용자 영역에 한글 중세 문자를 미리 디자인하여 넣어두고, 자신들이 사용하는 입력기를 이용하여 중세 문자에 해당하는 코드를 이 폰트에서 해당 글자가 차지하는 위치로 바꾸어주는 방식으로 처리하는 것이다.

입력기와 폰트 내의 문자를 대응시키려면 나름의 입력 방식이 있을 것인데 아직 필자는 그 방법이 어떤 것인지 들어보지 못하였다. 다만, 해당 문자의 PUA 코드값을 알기만 한다면 그 위치의 글자를 불러들여서 옛한글이 표현되는 것처럼 할 수 있다는 것만은 알고 있다.[8]

더 심각한 것은, 그나마 중세 문자를 포함하고 있는 한글 폰트로는 위에 열거한 폰트들이 유일하다는 것이다. 한컴바탕과 새바탕은 사실상 동일한 글꼴이며, 모두 한양시스템에서 OEM 제작한 것이라 한다.[9]국가가 많은 돈을 들여 제작했던 문체부 글꼴은 영문자와 기본 상징문자, 한자마저 갖추지 않고 오직 KS X 1001 완성형 한글 부분만 채워넣은 부실한 것이었고, 여러 글꼴 회사에서 판매하고 있는 상업용 글꼴들마저 유니코드 한글 영역

[8] 흔글이나 MS Office가 아니어도 사용자 영역 옛한글 문자를 입력할 수 있게 하는 프로그램으로 한적입력기가 있다고 한다.

[9] 한양 계열의 글꼴은 모두 트루타입인데, 한컴바탕, 한양바탕, 새바탕, sun바탕이 사실상 거의 동일하고, 한컴돋움, 한양돋움, 새돋움, sun돋움도 역시 거의 동일한 글꼴이다. MS Office Plus Pack에는 새굴림이라는 글꼴을 하나 더 볼 수 있고, 새궁서에 해당하는 한양해서도 유니코드 영역을 포함하고 있다.

조차도 갖추지 않은 상태로 출시되고 있는 것이 현실이다.

결국 문제는, 중세 문자 사용이 필요한 대부분의 상황에서 표준인 "첫가끝"이 사용되는 경우가 없으며, 중세 문헌 데이터베이스 구축에서도 "첫가끝"이 고려되는 경우를 찾아볼 수 없다는 데 있을 것이다.

2.2 Lambda용 한글 패키지

문화관광부의 위탁 연구 보고서인 듯한 [고기형1999]는 아마도 Omega로 한글을 처리하는 문제에 대한 최초의 문헌이 아닌가 싶다. 이 문서에 의하면 Unicode, EUC-KR, CP949를 모두 지원하는 OTP와 현대 한글과 확장 한자 전부를 표현할 수 있는 OFM이 제작되었다고 한다. 그리고 두 개의 OTP(`euckr-uni.otp`, `uhc-uni.otp`) 파일의 소스 일부가 소개되어 있다.[10] 그러나 어떤 이유에서인지는 알지 못하지만 한국과학기술원 밖에 있는 일반 사용자는 이 한글 Omega에 대해 접근할 수 없었다. 공개가 되어 있는 것은 hLaTeXp뿐이다.

현재 이용할 수 있는 Lambda용 한글 패키지는 HLaTeX의 hangul, u8hangul, 그리고 DHHangul의 dhhangul이 있다. 이 패키지들이 처리하는 문서의 입력 인코딩은 표 3.1과 같다.

표 3.1: 한글 Lambda 패키지

입력 인코딩	hangul	u8hangul	dhhangul
CP949	가능	*	가능
Unicode Hangul(UTF-8)	*	가능	가능
첫가끝 Unicode(UTF-8)	*	가능	가능
한양PUA 중세문자(UTF-8)	*	*	가능

특기해 두어야 할 것은 Lambda가 아니라 LaTeX을 사용하지만 한글 문자의 경우 UTF-8 입력을 받아들여 처리할 수 있는 CJKLaTeX의 `UTF8` 환경이다. Werner Lemberg 씨의 이 패키지는 특히 한양계열 폰트의 PUA 영역

[10] 이 OTP의 저자는 차재춘 님으로 되어 있다.

중세문자를 식자하기 위해 흔히 이용되곤 했다. 이 방법은 일찍이 KTUG에서 이미 시도된 바 있으며, 일본어로 된 TeXで朝鮮語を使う+PDFの作成이라는 흥미로운 사이트도 본 바가 있다.[11]

2.3 Lambda에서 한글 글꼴

확장 한자, 현대 및 중세 한글 문자만을 식자하려 해도 현실적으로 부딪치는 문제는 글꼴이다. 한글 트루타입 글꼴 사용 방법이 개발된 이후 글꼴 사용 자체의 기술적 문제는 거의 없어졌다 해도 과언이 아니다. 그러나 역시 문제는 "모든 한글" 문자나 확장 한자 문자, 중세 문자 또는 중세 문자 식자를 위한 한글 자소를 포함한 글꼴은 표 3.2에 보인 것이 거의 전부라 해도 과언이 아니다. 한양바탕과 한양돋움 등은 글자의 갖춤은 거의 같다. 한양 obatang과 한양 ogulim도 마찬가지이다. 은글꼴에서 [U+1100] 자소를 갖춘 것은 은바탕(UnBatang-odal.ttf)이다.[12]

표 3.2: Lambda에서 이용할 수 있을 한글 글꼴

글꼴	윈도바탕	은바탕	한양바탕	옛바탕
KSX1001 한글	O	O	O	*
Unicode 한글	O	O	O	*
KSX1001 한자	O	O	O	*
KSX1002 한자	X	X	O	*
Unicode CJK 공통한자	O	X	O	*
Unicode CJK 확장한자	X	X	O	*
PUA 중세문자 및 구결	X	X	O	*
[U+1100] 자소	X	O	X	O

그러므로, 만약 작성하려 하는 문서가 "현대 한글"만으로 이루어져 있다면 hangul 패키지를 이용하고 CP949 인코딩의 문서를 만들어서 은글꼴을

[11] 이 문서에서 완성형 이외의 문자를 식자하기 위하여 사용한 hlatexcjk 패키지는 바로 이 환경을 이용하는 것이다.

[12] 백묵글꼴의 사정이 어떠한지 알 수 없었다.

사용하여도 된다. 그러나 확장 한자나 중세문자를 포함하고 있다면 어쩔 도리없이 한양 글꼴을 고려하지 않을 수 없는 것이 (매우 불만스러운) 현실이다.

DHHangul은 위에 열거한 글꼴을 거의 모두 사용할 수 있고, 중세 문자의 표현에서도 "첫가끝" 한글과 한양 PUA 중세 문자를 모두 처리할 수 있다. 현재 가능한 모든 한글 식자 방법의 최대한을 구현한 Lambda 패키지라고 생각된다.

제 3 절 ksx1001-k

hangul-k 패키지는 LaTeX 패키지이다. 그러므로 직접적으로 Lambda를 지원하는 것은 아니다.

HLaTeX의 hangul 패키지는 Lambda가 실행되면 ksx1001.tex이라는 패키지를 부르도록 되어 있다. HLaTeX–Lambda란 다름아닌 `ksx1001.tex`이다.

원래 KS X 1001(KS C 5601) 한글만을 처리할 목적으로 작성된 때문이었는지 이 패키지의 이름이 `ksx1001.tex`이지만 앞서 설명한 대로 현재 이 패키지는 CP949 한글을 모두 처리할 수 있다.

hangul-k 패키지도 마찬가지 방법으로 동작한다. 즉, CP949 인코딩의 입력파일을 잘 처리한다. hangul-k 패키지가 부르는 것은 `ksx1001-k.tex`이고, 이것은 본질적으로 작동 방식은 ksx1001.tex과 동일하다. 단지 hangul-k 패키지와의 호환성을 위한 명령이 추가되었을 따름이다.

ksx1001-k.tex에서 자동조사는 실제 "자동"으로 이루어진다. 자동조사가 만들어내는 수많은 문제들은 LaTeX이었기 때문에 발생한 것이었음을 확인할 수 있다.

윈도 사용자는 간단히 다음과 같은 Preamble을 가진 입력 파일을 작성하고 컴파일만을 Lambda로 하는 것이 "모든 한글"을 사용하는 가장 좋은 방법이 될 것이다. 이것은 거의 전적으로 CP949 한글을 일상적으로 사용

하는 윈도 사용자를 위한 것으로, 리눅스 시스템에서 작업하거나 "진정한" 한글 Omega 문서를 만들고 싶다면 DHHangul 사용을 고려해볼 것을 추천한다.

```
\documentclass{article}
\usepackage{hangul-k}
\usepackage[dvipdfm,bookmarks=false]{hyperref}
\usepackage{unttf}
```

한글 문서를 Lambda용으로 작성하기로 결정하였다면 다음과 같은 점에 주의하라.

1. PDF의 한글 책갈피(bookmarks)는 만들 수 없다.

2. 글자가 표현되지 않는 것은 Lambda만의 잘못이 아닐 수 있다. 은글꼴이나 한양글꼴 이외의 글꼴을 사용하려 한다면 완성형 이외 문자가 표현되지 않는 것은 글꼴을 선택한 사용자의 책임이다.

3. LaTeX 패키지 가운데 어떤 것은 Lambda에서 완전하게 호환되지 않는 경우가 이따금 있다고 한다. 예컨대 framed 패키지의 일부 환경은 Lambda에서 뜻하지 않은 동작을 하기도 하고, 순전히 LaTeX만을 의식하고 만들어진 hlatexcjk는 Lambda에서 사용할 수 없다. LaTeX용 영문 글꼴 패키지들도 Lambda에서 사용되기 어려운 것이 많으므로 주의를 요한다.

4. LaTeX과는 다른 명령을 사용하는 경우가 있다. 예컨대 hangul-k 패키지의 글꼴 선택 명령 `\hfontfamily`와 같은 것은 `\fontfamily` 등으로 바꾸어 주어야 한다.

5. ksx1001.tex에서는 영문 글꼴은 한글 글꼴의 영문 영역을 사용한다. 그러므로 txfonts 등을 선택해도 효과가 발생하지 않을 것이다.[13]

[13] DHHangul은 영문 글꼴과 한글 글꼴 영역을 분리해놓았다.

제 4 장

PDF 문서만들기

이 장은 원래 독립된 글로 쓰여졌던 것이다. 그래서 앞 장에서 다루어진 주제와 겹치는 부분이 남아 있다. 그러나 "한글 PDF 문서 작성"의 일반적 가이드로서 같은 내용이라도 그 맥락이 동일하지 않으므로 의미가 없지 않다고 생각되어 그대로 두기로 하고, hangul-k 패키지에 대한 부분만 조금 줄였다.

제 1 절 한글 PDF 문서 작성: 들어가는 말

KTUG을 중심으로 한글 PDF 문서를 작성하는 방법이 발전하였다. 그러나 아직까지 쉽게 읽을 만한 안내문서가 없어서 KTUG 문서화 프로젝트 팀에서 이 문서를 준비하였다. 이 장은 KTUG에서 제공하는 hangul-k, hlatex-interword, hsectsty, hsetspace, myulem 패키지를 이용하여[1] 고품위의 한글 PDF 파일을 제작하는 방법에 대해서 간략하게 설명한다. 관련된 주제들이

[1] 이 글은 여기 열거된 h-시리즈 패키지의 사용설명서를 겸한다.

너무나 방대하고 경우에 따라서는 매우 전문적이기 때문에, 최종 사용자의 입장에서 당장 적용해볼 수 있는 "방법"의 제시에 중점을 두었다.

LaTeX에서 PDF를 얻는 일반적인 방법에 대해서는 KTUG의 FAQ를 참고하라.[2]

여기서는 Lambda(Λ)나 PDFLaTeX을 사용하지 않고 LaTeX으로 얻은 DVI 파일을 DVIPDFMx로 PDF 변환하는 방법을 사용하겠다. 한글 문자가 중심인 한글 문서를 작성하는 데 있어서는 이 방법이 가장 효과적이고 사용자의 다양한 요구를 충족시켜 주는 것이기 때문이다.[3]

제 2 절 기본적인 사항

2.1 PDF의 목적: 인쇄인가 화면보기인가

PDF 문서는 온라인 문서의 한 표준이 되었다. 생각해볼 것은 사용자가 PDF로 최종결과를 얻으려 하는 목적이 무엇인가에 있다. 인쇄용으로 사용할 문서는 화면보기에서 좋은 결과를 얻는 것과는 다를 것이다. 최근 들어서 PDF를 출력용으로 사용하는 경우가 많아졌다고 한다. 따라서 PDF를 인쇄에 사용할 PostScript 대신 쓰기 위해서 작성하는 경우도 분명히 있을 것이다. 이 글에서는 인쇄만을 목적으로 하는 PDF에 대해서는 논외로 하려 한다.

인쇄에 사용할 목적이라면 모르지만 온라인 화면보기용으로 PDF를 작성한다면 주의해야 할 것이 몇 가지 있다.

① 텍스트 검색이 되어야 한다. 온라인 문서에서 원하는 단어나 어구를 찾을 수 없다면 얼마나 답답하겠는가?

[2] http://faq.ktug.or.kr/mywiki/

[3] 예컨대, DVIPDFMx를 이용하여야만 한글 책갈피(bookmarks)를 얻을 수 있다. 그리고 텍스트의 검색·추출이 가능한 PDF 문서를 얻을 수 있게 할 뿐아니라, 문서에 암호를 걸 수도 있다. 이런 모든 기능은 현재로서는 오직 DVIPDFMx만이 제공하는 기능이다. 물론 한글이 문제가 아닌 경우는 그렇지 않다.

② 하이퍼링크가 작동하여야 한다. 온라인 문서에 하이퍼링크가 없다면 진정한 의미에서 "온라인" 문서라 하기 어려울 것이다.

③ 책갈피(bookmarks)와 같은 문서내 탐색장치가 있으면 편리하다.

2.2 기본 전략

이 글이 채택하고 있는 문서 작업의 기본틀은 다음과 같다.

Compiler: LaTeX으로 컴파일하여 DVIPDFMx로 PDF 변환한다. PDFLaTeX 등을 사용하지 않는다.

Viewer: gv, xpdf 등을 사용하는 경우는 문제삼지 않는다. Adobe Reader 또는 Acrobat Reader로 화면보기하는 경우만을 고려하겠다.

Packages: HLaTeX-0.991과 hangul-k 패키지를 사용한다.

제 3 절 글꼴

3.1 글꼴 사용 일반

3.1.1 PDF에서 이용할 수 있는 글꼴의 종류

일반 사용자들은 글꼴에 예민할 수밖에 없다. 최종적으로 만들어진 PDF는 다양한 글꼴을 이용할 수 있는데, 어떤 글꼴을 사용하였느냐에 따라 사용자의 직관적인 만족도가 상당히 많이 달라지는 것을 볼 수 있었다. 한글 글꼴과 관련하여 다음과 같은 몇 가지 글꼴이 PDF에서 사용된다.[4]

비트맵 글꼴 Type 3 글꼴이라고도 한다. 기본적으로 점(pixel)의 집합으로 문자를 표현하는 글꼴이다. 이 글꼴의 해상도(resolution, dpi)가

[4] 글꼴 자체에 대한 전문적인 논의는 관련된 사이트나 책을 참고하라. 여기서는 PDF로 만들어진 파일을 화면 디스플레이할 때 사용자의 직관적인 관점에서 각 글꼴이 어떠한 차이를 보이는지만을 기술한다.

프린터의 해상도와 일치한다면 출력물의 품위가 크게 차이가 나지는 않지만 화면보기를 할 때는 특히 Acrobat Reader 5.x 버전 이하로 볼 때 글자윤곽이 찌글거려서 독자의 불만을 사는 글꼴이다. 다만 6.0 이후 버전은 비트맵 글꼴도 비교적 잘 표현해주는 것으로 생각된다. 이 글꼴은 사실상 글자라기보다 그림에 가깝기 때문에 "텍스트"로 다루어질 수 없고, 그 결과 검색이나 추출은 불가능하다. TEX에서는 원래 PK 글꼴이라 불리는 비트맵 글꼴을 TEX 자신의 METAFONT로부터 얻어내어서 인쇄와 화면 디스플레이에 사용했다.[5] 인쇄 결과는 매우 훌륭하지만 한글과 같이 수많은 폰트 파일이 필요한 경우 해상도에 따라 매번 PK 픽셀 글꼴 파일을 만드는 것은 좀 지루한 일이었고, PDF에 포함시킨 결과는 그다지 만족스럽지 못했다고 할 수 있다.

PostScript 글꼴 `*.pfb` 또는 `*.pfa` 확장명을 가지는 Type 1 글꼴이 대표적이다. 윤곽선 글꼴이며 PDF에 가장 잘 맞는 글꼴 형식인 듯하고, 화면보기의 상태도 좋다.

TrueType 글꼴 한글은 PostScript 글꼴이 아주 예외적으로만 사용되고 있다. 예컨대 전문 출력소에서 사용하는 글꼴은 PostScript 글꼴이다. 그러나 일반 사용자가 PostScript 글꼴을 대하기란 아주 어려운 일이고 따라서 보다 저렴한 TrueType이 널리 쓰이게 되었다. 대부분의 한글 글꼴은 TrueType 형태로 쉽게 구할 수 있다. 윈도 운영체제의 기본글꼴들도 TrueType이다.

OpenType 글꼴 TrueType과 PostScript 글꼴을 모두 포괄하는 차세대 글꼴이지만 한글 글꼴에 관한 한 아직은 그다지 많지 않다.

3.1.2 한글 PDF 만들기의 몇 가지 방법

한글 PDF가 만들어지는 일반적인 상황을 생각해보자.

[5] TEX의 폰트 사용방법에 대해서는 이 글에서 자세히 논의할 수 없다. 관련된 다른 자료들을 참고할 것.

워드 프로세서를 사용하는 경우. MS Word나 혼글과 같은 워드 프로세서에서 문서를 작성한 다음 Acrobat Distiller를 이용하거나 그와 유사한 PDF 인쇄 프로그램, 예컨대 ezPDF와 같은 프로그램으로 인쇄하여 PDF를 얻어낼 것이다. 윈도 운영체제에서 실행되는 워드 프로세서라면 글꼴도 TrueType이 사용되었을 것이므로 PDF 변환 프로그램을 어떻게 설정했느냐에 따라 달라지기는 하겠지만 TrueType을 내장(embed)하거나 하지 않은 PDF가 만들어질 것으로 생각된다. 이것이 뜻대로 되지 않으면 아마도 Type 3 비트맵 폰트가 들어갈 것이다. 그러나 이 결과에는 사용자가 그다지 만족하지 않을 것 같다.

PostScript 인쇄 결과를 이용하는 경우. 보다 일반적인 PDF 작성 방법은 PostScript 출력(`*.ps` 또는 `*.prn`)을 적당한 PostScript 해석기(예를 들면 GhostScript)로 PDF 변환하는 것이다. 이 때는 원래의 PostScript 출력에 어떤 글꼴이 어떻게 사용되었느냐가 문제인데, 매킨토시의 QuarkXpress의 경우를 예로 들어 보자. QuarkXpress에서는 작업시에 화면용 글꼴이라는 것을 주로 사용하는데 이것은 같은 이름의 PostScript 글꼴을 인쇄에 사용할 것을 전제로 draft용으로만 사용하는 비트맵 글꼴이다. 이 경우 화면상의 작업용 글꼴을 내장(embed)해서는 아니될 것이므로 일관되게 글꼴을 내장하지 않는 PostScript 파일을 만들어내고 있다.

LaTeX에서. LaTeX 사용자는 dvips에 의하여 PostScript 출력 파일을 얻을 수 있다. 이것을 예컨대 GhostScript의 ps2pdf 스크립트를 이용하여 PDF로 변환할 수 있다. 그런데 dvips는 (아직) TrueType이나 OpenType을 처리하는 방법을 제공하지 않는다. 따라서 TrueType, OpenType은 PK 비트맵 픽셀 폰트 파일로 변환되어(이 때 ttf2pk와 같은 유틸리티가 사용된다.) PostScript 파일로 포함될 것이다.

DVIPDFMx나 PDFLaTeX은 TrueType과 PostScript 폰트를 모두 잘 지원한다. 그러나 한글 TrueType 글꼴 사용에 있어서 PDFLaTeX은 몇 가지 한계를 가지고 있다. 이것은 PDFLaTeX 자체의 문제라기보다 PDFLaTeX을

위한 한글 TrueType 사용 기술이 충분히 개발되지 않았기 때문이다. 기울인 글꼴(oblique)을 사용할 수 없고 텍스트 검색이 되지 않는다.

3.2 한글 PDF의 글꼴 사용 선택

3.2.1 한글 TEX 시스템들의 글꼴 사용 문제

hLaTeXp는 비록 METAFONT source를 제공하지 않았지만 매킨토시의 글꼴이었던 신명조, 화명조 글꼴(PostScript)로부터 추출된 PK글꼴을 제공하였다. 그리고 혼TeX은 윈도 TrueType 글꼴을 이용하였다. 이 두 프로그램은 사실상 거의 동일한 TeX 엔진을 탑재하고 있었는데 최종 출력된 글꼴의 품위는 어느 정도 사용자에게 만족을 주는 수준이었다고 할 수 있다. 특히 혼TeX으로부터 Acrobat Distiller를 이용하여 만들어낸 PDF 파일은 트루타입을 내장할 수 있었으므로 워드 프로세서 혼글로부터 ezPDF를 이용하여 얻은 PDF와 큰 차이가 없었다.

HLaTeX은 0.99 이후로 UHC 글꼴을 채택하고 있다. 이것이 기본글꼴이고 PDF로 만들었을 때는 PostScript이므로 비트맵 글꼴에서 얻어진 불만족스러운 결과를 줄일 수 있었다. 그러나 UHC 글꼴은 아무래도 상업용 글꼴이 익숙한 워드 프로세서 사용자들에게는 조금 "촌스럽게(?)" 보이는 점도 없지 않았던 모양이다.

결과적으로, hLaTeXp는 글꼴의 디자인 품위는 나쁘지 않았지만 PK 비트맵 픽셀 폰트밖에 이용할 수 없었으므로 고품위 PDF를 만드는 데 적합하지 않았고, 혼TeX은 윈도 TrueType을 이용하였으므로 글꼴의 품위나 PDF 출력결과는 괜찮았지만 프로그램 자체가 치명적인 문제점을 지니고 있었고, HLaTeX은 PostScript 한글 글꼴을 제공한다는 점이 매력적이었으나 글꼴의 디자인에 대해 불만족한 사용자가 일부 있었다는 정도로 정리할 수 있겠다.[6]

[6] UHC글꼴의 디자인에 불만인 사용자가 있었다는 것은 게시판에 올라온 몇몇 글에서 유추한 것으로, 아마도 PDF 상에서 자소조합 때문에 시작적으로 조금 위치가 어긋나 보이는 것을 참아내지 못한 "습관의 문제"일 가능성을 배제할 수 없다.

이 모든 문제를 이제는 HLaTeX 위에 한글 TrueType을 이용함으로써 극복할 수 있게 된 셈이다.

3.2.2 UHC 폰트와 트루타입 폰트

우리는 한글 PDF가 문제이므로, 결국 다음과 같은 두 가지 가운데 하나를 선택해야 한다.

① PostScript Type 1 글꼴인 UHC 글꼴을 사용하는 것. 이 때는 글꼴이 내장된다.

② 한글 TrueType 글꼴을 사용하여 글꼴이 내장되거나 되지 않은 PDF를 만드는 것.

첫번째 방법을 사용하게 되면 UHC 글꼴의 특성을 고려하지 않을 수 없다. 은광희 님의 UHC 글꼴은 현재 공개되어 있는(GNU GPL 라이센스) 유일한 PostScript 한글 글꼴이다. 그런데 이 글꼴은 자소조합 글꼴이라는 특징을 가지고 있다. 화면상으로 글자들이 조금 비뚤비뚤하게 보이는 이유는 이러한 특성 때문이다. 그리고 PostScript 인쇄기가 Font Caching을 하는 경우에는 일부 캐시된 글자들이 뭉개지는 문제가 있는 것으로 보고되어 왔다. 해결방법이 없는 것은 아니지만 프린터의 설정을 일일이 맞추도록 요구하는 것은 쉬운 일이 아니었으므로, 그런 불편을 감수해야 한다. 또, 자소 조합 글꼴이기 때문에 이 글꼴을 내장한 PDF 파일에서는 검색과 추출이 불가능하다. 이 때문에 UHC 글꼴로 만든 PDF 파일들에 대하여 인쇄상의 문제를 호소하는 경우가 상당히 많았다.

트루타입을 이용하는 것이 현재로서는 최선의 선택이 될 성싶다. 다만 트루타입은 LaTeX이나 HLaTeX 자체에서 기본으로 제공해주지 않는 것이므로 사용자가 자신에게 맞게 설정하여 사용해야 한다.

3.3 트루타입 폰트 설정

TrueType 글꼴로 PDF를 만들기로 작정하였다면 TrueType을 LaTeX에서 사용할 수 있도록 설정해주어야 한다.

미리 만들어진 폰트 패키지. KTUG에서는 TrueType 폰트 사용 기술이 집중적으로 발달하였다.[7] 그 작업의 파생물로서, TEXMF TREE의 형태로 묶어져서 배포되는 TrueType 폰트 패키지들이 만들어졌다. 대표적인 것이 아시아폰트 기본팩이며, 윈도 기본글꼴 사용을 위한 패키지, 문화부 글꼴 패키지 등이 있다. 이 TrueType 글꼴 패키지들은 KTUG FAQ 사이트에서 찾을 수 있다.

압축파일 형태로 제공되는 글꼴 패키지들을 내려받아서 새로운 TEXMF TREE로 등록하고[8] `dvipdfmx.cfg`에 `map` 파일을 추가해주는 것으로 이 글꼴들을 사용할 준비는 대부분 끝난다. 한 가지 예만 들어두겠다. 이 과정은 자신이 사용하는 TeX 배포판에 따라 달라진다는 점을 기억하자.

아시아폰트 기본팩을 다운받아서 `C:\texmf-asiafonts-basic`에 풀어놓고 Filename Database를 갱신하였다면,

```
#> kpsewhich --format="other text files" \
       --progname="dvipdfm" dvipdfmx.cfg
```

이 명령으로 `dvipdfmx.cfg`의 위치를 확인하여 그 파일을 연 다음 마지막에

```
f asiafonts-basic.map
```

이 한 줄을 추가해준다.

---

[7] 주로 조진환(ChoF) 님의 DVIPDFMx 작업이 이러한 결과를 가져왔다. 여기에 기여하신 분들로는 박원규, 김도현, 신정식 님 등이 있다.

[8] 이 작업은 TeX 배포판에 따라 하는 방법이 다르다. MiKTeX의 경우에는 MiKTeX Options를 실행하여 `Roots` 탭에서 원하는 TEXMF TREE의 root를 찾아서 추가하고 `Refresh Now`를 실행하면 된다.

특히, 박원규 님에 의하여 UHC 글꼴이 TrueType으로 포팅되었는데 이것을 은글꼴이라 한다. 은글꼴도 위와 같은 패키지 형식으로 배포된다. UHC 폰트를 좋아하는 분이라면 이 글꼴을 채택하는 것을 권장한다.

사용자 자신의 폰트 설정 김도현 님은 TrueType 설정을 보다 쉽게 하는 TTF2HLaTeXFont라는 perl script를 제작하였다. 이 스크립트를 사용하면 훨씬 자유롭게 여러 트루타입 글꼴을 자신의 문서에서 사용할 수 있다. 이 스크립트의 사용법은 아주 간단해서 몇 가지 폰트 결합을 config 파일에 지정하고 실행하는 것으로 끝이다. 심지어 TEXMF Tree의 형태로 설치해 주는 일까지 해준다. 사용방법은 24페이지나 스크립트 배포처를 참고하라.

DVIPDFMx를 사용하면 한글 TrueType 글꼴을 내장(embed)하지 않도록 설정할 수 있다.[9] TEXMF TREE 묶음 형태로 배포되는 글꼴 패키지 가운데 mskttfonts라는 윈도 기본 글꼴 사용 패키지는 기본값이 글꼴을 내장하지 않도록 설정되어 있다. 이 방법의 장점은 만들어진 PDF의 크기가 정말 작다는 것이다. 그러나 사용자의 컴퓨터에 그 문서에 사용된 글꼴이 이미 설치되어 있어야 동일한 모양으로 디스플레이된다는 점이 한계이다. 한글 윈도 기본글꼴을 이용하는 위의 패키지로 작성된 문서는, 한글 윈도 기본글꼴이 없는 환경에서는, 만약 Adobe Reader가 한국어 패키지를 설치하고 있는 상태라면 기본 글꼴로 대치해서 화면에 보여주기는 하겠지만, 동일한 상태로 보여진다고 장담할 수는 없다.

요컨대, 글꼴 내장하는 방법은 화면보기와 인쇄에서 언제나 동일한 형태를 유지할 수 있지만 파일크기가 커지고, 글꼴을 내장하지 않으면 파일의 크기는 작아지지만(그러나 많은 그림을 포함하고 있다면 큰 차이가 나지는 않을 수도 있다), 폰트를 가지고 있지 않은 사용자에게서 열리지 않을 가능성이 있다는 것이 문제이다.[10]

---

[9] 이 설정은 `map` 파일을 수정하면 된다. 구체적인 방법에 대해서는 다른 문서를 참고하라.

[10] 최근의 Adobe Reader 6.0 이후 버전은 필요한 파일을 인터넷에서 다운받아서 설치

이로써 글꼴 결합이나 설정이 자유로와졌지만 자유에는 책임이 따르는 법이다. 과도하게 많은 글꼴을 무분별하게 씀으로써 문서의 품위를 낮추고 가독성을 떨어뜨리는 것은 전적으로 사용자 자신의 책임이다.

참고로, 이 문서는 이 분야에서는 이제 고전이 된 "HLATEX에서 임의의 한글 TrueType 글꼴 사용하기"라는 문서에서 제시된 글꼴 결합을 기초로 만들어졌다. 이 문서는 TTF2HLaTeXFont가 나오기 이전에 작성된 것이므로 굳이 따라하여 볼 필요는 없다.

마지막으로 글꼴의 설정과 사용에 대해서 추가할 것이 있다. 일반적인 교훈은 "하나의 문서에 세 종류 이상의 글꼴을 사용하지 말라"는 것이다. 그 세 종류란, 본문글꼴(\rm), 산세리프글꼴(\sf), 타입라이터(모노스페이스) 글꼴(\tt)를 가리킨다고 할 때, 적당한 한글 글꼴을 여기에 대응시켜서 사용하는 것이 논리적으로 일관성이 있는 것일 터이다. 필요하다면 강조(\emph) 명령에 쓰일 글꼴(영문에서는 이탤릭 글꼴)을 별도로 설정할 수 있을 것이다.

3.4 영문 및 수학 글꼴과의 관계

한글 문서를 작성할 때는 한글 글꼴의 선택도 중요하지만 영문 글꼴도 주의깊게 선택하여야 한다. 그 이유는 HLATEX이 한글 문서의 문장부호와 숫자에 영문 글꼴을 적용하고 있기 때문이다. 그 결과 어떤 영문 글꼴을 선택하였느냐에 따라 문서의 느낌이나 품위가 달라지게 된다.

수식을 많이 쓰는 문서를 작성한다면 수학 글꼴에도 고민을 좀 해야 할 것이다. 한글 본문 글꼴과 가장 잘 어울리는 영문 글꼴은 어떤 것이 있는지 경험을 통해서 직접 자신의 취향에 맞는 것을 찾아가는 도리밖에 없다.

가장 쉬운 선택은 "선택을 하지 않는 것"이다. 이 경우에는 Computer Modern 글꼴이 사용될 것인데, PDF 제작을 위해서는 되도록 Type 1 Scal-

하고 보여주는 기능을 내장하고 있다. 그러나 누구나 Adobe Reader를 쓰는 것도 아니며, 어디서나 6.0 이후 버전이 설치되어 있는 것도 아니고, 항상 인터넷에 연결되어 있지 않은 곳도 있는 것이다.

able CM 폰트를 사용하는 것이 좋을 수도 있을 것이다. 이 경우 Type-Writer 서체는 아주 훌륭하고, 수학 글꼴과의 조화도 고민할 필요가 없다.[11]

영문 글꼴과 수학 글꼴에 대해서는 KTUG FAQ의 TeX 글꼴 페이지와 수학용 글꼴 페이지에 읽을 만한 정보가 있다.

이 문서에서 사용하고 있는 영문 글꼴은 **Utopia 글꼴**이다. 위의 사이트에서 관련된 정보를 얻을 수 있을 것이다.

제 4 절 하이퍼링크와 한글 책갈피(bookmarks)

하이퍼링크와 한글 책갈피는 PDF 문서를 풍부하게 만드는 요소이다. hyperref 패키지를 사용하여 구현한다. 주의할 것은 hyperref 패키지를 로드할 때 `[dvipdfm]` 옵션을 주라는 것이다. 안타깝게도 아직까지 `[dvipdfmx]`라는 옵션은 없지만 DVIPDFMx를 사용하는 것을 명시해두는 것이 좋다.

4.1 하이퍼링크

그 이후, 하이퍼링크는 패키지가 제공하는 기능을 써서 구현한다. 예를 들어 KTUG 사이트로 외부 하이퍼링크를 걸려면

```
\href{http://www.ktug.or.kr}{KTUG}
```

와 같이 하면, KTUG에서 보는 것처럼 외부 하이퍼링크가 만들어진다.

hyperref의 다양한 기능을 이용하면 하이퍼링크된 텍스트의 형태나 색상 등을 조절할 수도 있고 그밖에도 여러 가지 재미있는 기능을 구현할 수 있다. 더 자세한 것은 hyperref의 패키지 문서나 다른 LaTeX 관련 문서를 참고하라.

[11] 최근의 대부분의 배포판은 DVIPDFMx 또는 PDFLaTeX을 위한 폰트 정의의 기본값으로 METAFONT CM이 아닌 Type 1 CM을 사용하도록 설정해두고 있다. 그러므로 PDF를 만드는 상황에서는 Type 1 CM에 특별히 신경쓸 이유가 없는 경우가 많다. 다만, dvips를 이용하여 PS 파일을 먼저 만드는 경우에는 `-P pdf` 옵션을 지정하면 된다.

4.2 책갈피

책갈피(bookmarks)는 Adobe Reader 또는 xpdf 프로그램에서 지원하는 화면 디스플레이 기능의 하나이다. hyperref 패키지는 이 책갈피를 자동으로 만들어준다. 그렇지만 LaTeX으로 한글 책갈피를 넣는 일은 오랫동안 해결되지 않았던 문제였다. hyperref이 한글이 포함된 문자를 책갈피로 처리하는 데 어려움을 겪었기 때문이다.

한글 책갈피 만들기는 전적으로 DVIPDFMx의 기능이다. 따라서 예컨대 PDFLaTeX이나 GhostScript의 ps2pdf 등으로는 이런 결과를 (아직) 얻을 수 없다. 그리고, 한글 책갈피가 오류 없이 작동하려면 원칙적으로 한글 윈도 운영체제 상에서 한글 Adobe Reader를 사용하여야 한다. 다른 상황에서 되는 경우도 있겠지만, 여기서는 문제삼지 않겠다.

한글 책갈피 설정을 위해서는 Preamble에 다음과 같이 써넣고 컴파일하는 것으로 충분하다. 나머지는 DVIPDFMx가 알아서 해준다.

```
\usepackage[dvipdfm,CJKbookmarks]{hyperref}
\AtBeginDvi{\special{pdf: tounicode KSCms-UHC-UCS2}}
```

책갈피를 만들지 않으려면 다음과 같이 하라.

```
\usepackage[dvipdfm,bookmarks=false]{hyperref}
```

이 책갈피 만들지 않기는 DVIPDFMx가 나오기 전 한글 책갈피 때문에 어려움을 겪던 시절에 권장되던 방법이다. 이렇게 하지 않으면 컴파일 과정에서 무수한 경고가 쏟아져나오고 한글이 "깨진" 책갈피가 만들어지곤 했었다.

이 당시의 또다른 해결책으로 bmsec 패키지를 이용하여 한글 문서에서 영문 책갈피를 만드는 방법도 있었는데, 이것은 `\chapter`, `\section`와 같은 장절명령에서 특별한 옵션 인자로 영문 책갈피 문자열을 따로 지정해주는 방식이었다. 이 방법은 지금도 필요한 경우가 없지 않을 것이다.

제 5 절 문서의 레이아웃

5.1 종이 크기와 여백

5.1.1 종이 크기의 선택

종이 규격 페이지에 보면 인쇄에 쓰이는 종이 규격에 대한 설명이 있다.

온라인 문서라 하더라도 인쇄를 의식하지 않을 수 없으므로 익숙한 규격 용지에 맞추어서 작성하는 것이 나쁘지 않은 선택이다. 예컨대 A4 사이즈나 B5 사이즈는 전형적인 종이 규격에 해당할 것이다. 그런데 때로는 규격 외 사이즈로 문서를 만들어야 할 때가 있다.

LaTeX의 레이아웃 기본값은 전체적으로 보아서 상당한 기준의 요구를 충족시킨다. 그러나 여백은 많은 사람들이 너무 과도하게 주고 있는 것이 아니냐는 느낌을 받는 듯하다. 예를 들어 `[a4paper,10pt]`의 값으로 문서를 조판하면 `\textwidth`는 345포인트 정도가 된다. 이 값은 약 45mm 정도의 `hmargin` 값을 준 것과 비슷하다. 이 정도의 넉넉한 여백이 있어야 `\textwidth`가 지나치게 길어지지 않고 가독성을 높일 수 있다. 여백을 줄이고자 할 때는 여백을 줄인 결과가 독자의 독서를 방해하지는 않을 것인지 한 번 더 생각해보기 바란다.

종이 크기와 텍스트 영역은 서로 밀접한 관계가 있다. 이것은 geometry 패키지를 이용하여 쉽게 구현할 수 있다. 예를 들면 이 문서는 그림 4.1과 같은 방법으로 종이 크기와 텍스트 영역을 설정하였다. layouts 패키지를 이용하면 현재의 디자인 상태를 좀더 잘 확인할 수 있다. layouts를 이용하여 이 문서를 점검한 결과에 대해서는 그림 4.2를 참고할 수 있다.

5.1.2 비규격 종이크기의 PDF 만들기

온라인 문서를 작성하면서 선택할 수 있는 좋은 방법은 A4 사이즈보다 종이 자체의 크기를 작게 하는 것이다. 그러면 여백을 줄이면서도 판면의 가로 길이(`\textwidth`)를 지나치게 늘리지 않을 수 있을 것이다.

```
%% PaperSize Setting.
\usepackage[paperwidth=150mm,
    paperheight=212mm,
    hmargin=20mm,
    top=25mm,
    bottom=27mm]{geometry}
\setlength\headheight{27.5pt}
```

그림 4.1: 이 문서의 레이아웃 설정

이런 식으로 새로운 크기를 정했을 때, PDF 문서의 크기를 정해주려면 몇 가지 신경쓸 점이 있다.[12]

먼저, 인쇄되는 종이는 규격 용지로 하고 사용자가 설정한 (그보다 작은) 판면 설정을 다만 표시만 해주는 방법이 있다. 예를 들면 A4 사이즈의 종이에 이 문서를 찍으면 어디까지가 설정된 페이지의 경계가 되는지를 frame으로 그려준다든가 하는 방법이다. 이 목적을 위해서는 crop 패키지가 쓰인다.

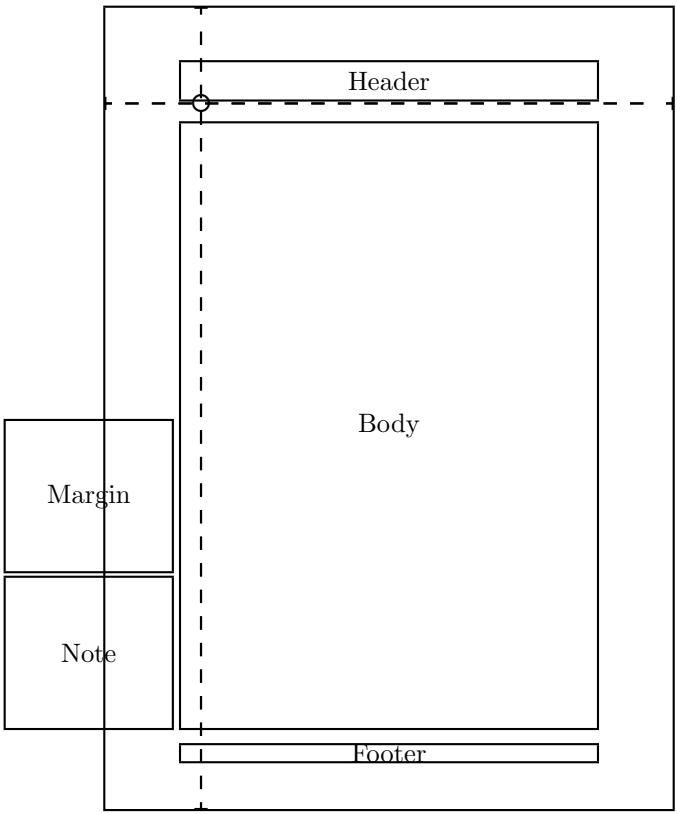
또다른 방법은 PDF 자체의 크기를 설정된 페이지에 맞추는 방법이다. 결과는 geometry로 설정한 `paperwidth, paperheight` 값에 따라 PDF 문서가 만들어지게 될 것이다.

만약 DVIPDFMx 최신 버전(20040320 이후 버전)을 사용한다면, 이 문제는 보다 쉽게 해결된다.[13] `-p` 옵션으로 종이 크기를 지정해주는 것이다. DVIPDFMx는 수많은 기본 종이값을 알고 있으므로 그 가운데서 선택하여도 좋고, 아니면 직접 가로x세로 길이를 지정해도 된다.

```
#> dvipdfmx -p "150mm,212mm" foo
```

[12] PDFLaTeX은 geometry 패키지를 읽어서 geometry가 설정한 종이와 판면 크기를 그대로 PDF로 만들어주기 때문에 매우 편리한데 DVIPDFMx는 별도의 절차가 필요하다.

[13] 이 기능이 추가된 최신 버전의 MiKTeX 바이너리 실행파일은 아직까지 구할 수 없다. 앞으로 나오겠지만 현재로서는 W32TeX에서만 윈도용 바이너리를 볼 수 있다. 유닉스나 리눅스 또는 CygWin 사용자라면 소스를 내려받아서 컴파일–빌드하여 사용할 수 있을 것이다.



Lengths are to the nearest pt.

page height = 603pt	page width = 427pt
\hoffset = 0pt	\voffset = 0pt
\evensidemargin = -15pt	\topmargin = -31pt
\headheight = 28pt	\headsep = 18pt
\textheight = 455pt	\textwidth = 313pt
\footskip = 25pt	\marginparsep = 7pt
\marginparpush = 5pt	\columnsep = 10pt
\columnseprule = 0.0pt	

그림 4.2: layouts 패키지로 그려본 이 페이지의 디자인

그러나 최신 버전이 아닌 이전 버전의 DVIPDFMx를 쓰고 있다면 이 때는 dvipdfm에 적용하던 방법을 이용한다. 문서의 첫 부분에 다음과 같은 명령을 포함한다.

```
\AtBeginDvi{%
  \special{pdf: pagesize width 150mm height 212mm}}
```

그리고, 반드시 geometry 설정을 hyperref 설정보다 앞에 두도록 하고, hyperref 설정 이후에 다음 그림 4.3의 코드를 포함한다.

```
\makeatletter
\AfterBeginDocument{%
  \ifx\special@paper\@empty\else
    \ifHy@setpagesize
      \special{papersize=\special@paper}%
    \fi
    \Hy@DisableOption{setpagesize}%
  \fi
}
\makeatother
```

그림 4.3: 임의의 종이 크기를 설정하기 위한 코드

이렇게 하면, 명령행에서 `dvipdfmx` 만을 실행하여도 적당한 크기의 PDF가 만들어지도록 할 수 있다.

5.2 글자 크기와 장평

영문의 경우와 달리 한글 글자는 모두 네모꼴이라서 글자의 가로폭이 비교적 균일하다는 특징이 있다. 영문의 경우는 variable width(가변폭) 글꼴이 대부분 본문 글꼴로 쓰이고 있기 때문에 커닝(kerning)이라든가 스페이싱이 매우 정교하게 발달하였지만, 한글 문서의 경우는 이 분야에서 그다지 많은 연구가 이루어지지 않았다.

활판 인쇄된 책들을 보면 글자는 네모꼴의 디자인 영역 안에 비교적 작게 들어앉고 글자와 글자 사이에는 조금 넉넉한 여유분이 있다. 그러나 전자조판된 책들은 글자와 글자 사이가 거의 여유가 없을 정도로 붙고 그 대신 단어와 단어 간격은 이전보다 더 벌어져 있는 것을 육안으로 확인할 수 있을 것이다. 이 "마이너스 자간"의 관행은 전자출판에서는 거의 확립된 조판 방식이 되었지만 그 효과가 과연 바람직한가에 대해 신뢰할 수 있는 통계적 증거가 아직은 없다고 생각된다. 즉, 글자와 글자를 붙인다고 해서 가독성이 높아지는 것도 아니라는 것이다. 단어별로 책을 읽게 되는 것이 아니라 일정한 어구를 한꺼번에 파악하면서 읽는 것이 일반적인 우리나라에서 문자폭의 절반 가까운 단어간 간격을 주는 것이 꼭 바람직한지는 잘 알 수 없다. 예를 들어 띄어쓰기에 있어서도 의존명사 앞에 오는 스페이스를 어절이 끝난 다음의 스페이스보다 더 적게 주는 것이 가독성을 높여주는 것은 아닐까? 아무튼 이런 점에 대해서 좀더 연구가 이루어졌으면 좋겠다.

TTF2HLaTeXFont 스크립트로 `*.ttf`로부터 `*.tfm`을 추출하면 `c` 시리즈 글꼴을 정의해준다. LaTeX NFSS의 `c` 시리즈 글꼴은 장평 75%의 글자체를 의미하는데, HLaTeX의 UHC 글꼴에도 이 시리즈가 정의되어 있다. 그러나 TTF2HLaTeXFont의 `c` 시리즈 글꼴은 장평 92%의 글자체를 의미하는 것으로 바뀌어 있다.[14] 한글에 대해서만 이 시리즈의 글꼴을 사용하도록 하려면 `\makethinhangul`이라는 명령을 선언해주면 된다.[15] 이 명령이 하는 기능은 `\hfontseries{c}`를 선언해주는 것뿐이다.

내부적으로 글꼴을 새롭게 설정하는 명령들이 있다. 예를 들면 각주나 그림과 표의 캡션, 장절명령, 색인(찾아보기)을 만드는 `theindex` 환경, 표지를 만드는 `titlepage` 환경이 대표적이다. 이러한 환경 또는 명령 안에서는 `\makethinhangul`이 작동하지 않으므로, 만약 이러한 곳에서도 `c` 시리즈 글꼴이 사용되기를 원한다면 거기에 맞추어서 `\makethinhangul`을 선

[14] KTUG의 정기 모임에서 이런 방식의 `c` 시리즈 글꼴을 채택하자는 비공식 논의가 이루어진 바 있었고, 최초로 시도된 것은 아시아폰트 기본팩에서였다.

[15] `\makethinhangul` 명령은 LaTeX이나 HLaTeX의 기본명령이 아니고 TTF2HLaTeXFont 스크립트에 의하여 생성되는 `xxttf.sty`에 정의되어 있다.

언해주면 된다. 이 문서에서는 예컨대 각주에서도 c 시리즈 글꼴을 사용하기 위해 다음과 같이 선언하였다.

```
%% footnote thinhangul setting.
\let\origfootnotesize\footnotesize\relax
\def\footnotesize{\origfootnotesize\hfontseries{c}}
```

c 시리즈 글꼴을 채택할 것이냐의 여부는 전적으로 개인적인 취향의 문제이다. 필자가 선호하는 것은 c 시리즈 글꼴을 사용하면서 본문을 11포인트로 하는 것이다. 10포인트에 92% 장평을 적용하면 글자가 조금 작아보이는 단점이 있는데, 화면용 PDF 파일의 작성에는 이 정도(11포인트 본문 + 92% 장평)의 설정이 가장 보기 좋다고 생각한다. 이밖에, 좀더 워드 프로세서에 가까운 느낌을 주려면 본문을 10.5포인트로 하는 방법도 있는데, 이것은 size10d5.sty라는 스타일을 사용해서 구현할 수도 있다.

5.3 행간과 자간

애초 HLaTeX은 행간이나 자간에 대한 고려가 거의 없었다. 영문 문서의 설정을 그대로 사용하게 되어 있었던 것이다. 그러나 타이포그라피의 측면에서 한글 문서의 행간과 자간은 영문 문서의 그것과 동일하면 문서가 지나치게 촘촘해 보여서 품위가 떨어진다.

hLaTeXp의 hangul 패키지는 기본 행간을 1.3으로 하는 한글 문서 설정 기본값을 제시했었다. hLaTeXp로 만들어진 문서가 그래도 읽기에 수월했던 이유가 아마도 이러한 기본 행간의 역할이 아니었을까 한다.

물론 행간은 사용자가 마음만 먹으면 얼마든지 쉽게 조절할 수 있다. 그러나 스타일의 기본값으로 제시되는 것이 있으면 아무래도 안심하고 쓸 수 있을 것이다. 사용자를 보다 편안한 문서 작성 환경으로 유도하는 효과도 있으리라고 생각한다.

5.3.1 자간과 단어간격

한글 문서에서 자간과 단어간격을 조절할 수 있게 하는 스타일 패키지는

hlatex-interword이다.

이 스타일은 *.dtx와 *.ins 형식으로 제공되므로 패키지 안내 문서를 hlatex-interword.dtx를 컴파일함으로써 얻을 수 있다. 간단히 필요한 것만 요약하면 다음과 같다.

[default] 옵션. 영문 문서의 경우와 동일한 자간을 사용하되 단어 간격을 조금 벌려주는 정도의 효과를 가져온다. 영문 문서 설정을 가져다 쓰는 경우보다 읽기에 조금 편안한 문서를 만들 수 있다. 이 문서가 이 옵션으로 작성되었다.

[HWP] 옵션. 흔글에서 작업할 때와 거의 유사한 단어 간격과 행나눔을 보여주는 옵션이다. 따라서 단어 간격은 상당히 넓은 편이다. 이 옵션을 더 좋아하는 분도 있으리라 생각한다.

[narrower] 옵션. 자간을 조금 촘촘하게 한다. "마이너스 자간"에 익숙한 분들이 좋아할 것으로 생각되고, 은글꼴을 본문 글꼴로 사용할 때는 꽤 좋은 효과를 가져온다. 한양 계열의 본문 글꼴에서는 어울리지 않는다는 것이 개인적인 생각이다. [default] 옵션과 함께 쓸 수 있다.

\interhchar 명령. 자간을 임의로 설정하게 해준다. 예컨대 다음 명령은 자간을 -1포인트 좁혀준다. 숫자는 0.1포인트 단위이다.

```
\interhchar{-10}
```

\interhword 명령. 단어간격 설정이다. 이 명령을 사용할 때 주의할 점은 남용하면 특히 문서 중간중간에 영어 문장이 들어가는 경우 너무 단어 간격이 넓어 보일 수 있다는 점이다. 자간은 영어 문장이나 문자에 영향을 끼치지 않지만 단어간격은 그렇지 않다.

세 개의 길이(dimension)를 인자로 취하는데, 마지막의 두 인자는 허용치의 상하한을 표시한다. 예를 들어

```
\interhword{.3em}{.2em}{.3em}
```

이 명령은 단어 간격을 0.5em을 기준으로 최대 0.7 (0.5+0.2)em, 최소 0.2 (0.5−0.3)em으로 조판하라는 지시가 된다.

아주 특별한 경우가 아니면 위의 임의 설정 명령들은 쓰지 않는 쪽이 좋다. 되도록이면 기본값으로 사용하도록 하자.

5.3.2 행간

행간은 보통 \baselinestretch 승수를 바꿈으로써 double spacing을 얻는 것이 일반적이었다. 영문의 경우 double spacing은 \baselinestretch 값을 1.6으로 하는 것이 권장되고 있다.

```
\renewcommand\baselinestretch{1.6}
```

또다른 방법으로 setspace 패키지가 쓰인다. 한글 문서 작성 상황에서도 이 패키지를 쓰는 것이 불가능할 것은 없지만, 이 패키지는 그 성질상 floating objects(떠다니는 개체)와 각주 안에서 모든 행간격을 1.0으로 되돌리는 기능이 있는데, 한글 문서의 경우 각주의 행간격이 1.0으로 강제되는 것은 무리라 하여 이것을 조금 유연하게 다룰 수 있도록 하고 한글 문서의 표준 행간격을 제시하기 위해 작성된 것이 hsetspace 패키지이다.

hsetspace에서는 몇 가지 추가적인 옵션과 명령을 제공한다.

[hangul] 옵션. 이 옵션을 주면 문서의 기본 행간격을 1.333으로 맞추고 각주와 floats, 그리고 quote, quotation 환경 내의 행간격을 1.2로 조절해준다. 만약 각주 사이의 간격을 조절하겠다면 [adjustfootnotesep]이라는 긴 이름의 옵션을 [hangul] 이후에 추가할 수 있다.

또, 여기에 [adjustverbatim] 옵션을 추가하면, verbatim 환경 내에서도 좁은 행간격(기본값은 1.2)을 적용해준다. 다만 이 모든 기능은 반드시 [hangul] 옵션을 준 상태여야 한다는 것이다.

[nofloatspacing] 옵션. floats와 각주에 적용되는 setspace의 기본 기능(행간격을 1.0으로 맞추는 기능)을 끈다. 그러므로 이 옵션을 준 이후에는 문서 내의 모든 행간격이 문서의 기본 설정 행간격을 따라간다. 예컨대 [hangul] 옵션을 선언하였다면, 각주 내에서도 행간격은 본문과 동일하게 설정된다.

그러므로, 이 옵션을 추가한 경우에는 [adjust...] 옵션은 무의미하다.

\SetHangulspace 명령. [hangul] 옵션을 준 경우에 사용할 수 있는 명령으로서 한글 문서에서의 행간격을 일괄 지정할 수 있게 한다. 두 개의 인자를 가지며, 첫번째 것은 문서 전체의 본문 기본 행간격이 되고 두번째 것은 floats와 각주 내의 행간격으로 쓰이게 된다.

```
\SetHangulspace{1.2}{1.0}
```

위의 명령은 본문의 행간을 1.2로 하고, floats, 각주, quote, quotation 내의 행간격을 1.0으로 설정하도록 한다. 여기서 숫자는 \setstretch 값, 즉 \baselinestretch 값이다.

각주 사이의 간격은 이것으로 조절되지 않으므로 \footnotesep 등은 사용자가 직접 지시하여야 한다.

이 이외의 다른 기능은 setspace의 경우와 같다. 즉, spacing 환경을 써서 문서 일부의 행간격을 임의로 조절할 수 있다. 그러므로 hsetspace와 setspace를 동시에 사용하지 않도록 하라. 둘 가운데 하나만 있으면 된다. setspace 패키지의 사용법에 대해서는 패키지 문서를 참고할 것.

5.4 장절명령의 설정

장절명령은 HLaTeX의 어려운 점 가운데 하나이다. HLaTeX의 hangul 패키지는 장(chapter)뿐 아니라 절(section)에도 "제 1 절"과 같은 형식으로 "제"와 "절"을 붙였는데, 이 때문에 적어도 장과 절에 대해서는 다른 영문 문서에서 적용되는 많은 장절형식 관련 패키지가 부적절하게 작동한다. 그

리고 간단히 \@startsection을 이용하는 영문 스타일의 절(section) 정의와도 잘 맞지 않아서 장절 형식을 바꾸고자 하는 사람들에게 많은 고통을 안겨주었다.

이 문제를 부분적으로 해결하고자 시도한 것이 hsectsty 스타일이다. 이 스타일은 sectsty을 한글화한 것인데 titlesec에 비해서 유연성이 조금 떨어지지만 그래도 장절의 폰트나 형식을 좀더 쉽게 바꿀 수 있다는 점 때문에 채용되었다. 장절명령에 대한 사항은 KTUG FAQ의 장절명령 페이지를 참고하기 바란다.

이 스타일은 기본적인 사용법이 sectsty과 같다. 그러므로 사용법 자체는 sectsty 패키지 문서를 참조하기 바란다. hsectsty를 로드하면 sectsty를 불러온다.

이 패키지를 사용하였을 때 기본값은 "제"와 "장" 또는 "절" 사이의 간격을 ~로 연결한 것보다 조금 좁게 잡아주는데, 이것이 마음에 들지 않아서 hangul의 원래 형식으로 되돌리려면 [THE] 옵션을 지정하면 된다.

추가된 옵션으로 [ensec]이 있다. 이 옵션을 주면 절(section)을 영문 문서의 절과 동일하게 식자해준다. 즉, "제"와 "절"이 붙지 않는다.

이 문서에서 hsectsty 패키지가 쓰였다. 관련된 코드를 보면 그림 4.4와 같다.

```
\usepackage{hsectsty} % 한글 장절명령 스타일.
\sectionfont{%
  \noindent\hfontseries{bc}\textcolor{blue}}
\subsectionfont{%
  \noindent\hfontseries{bc}\textcolor{DarkMagenta}}
\subsubsectionfont{%
  \noindent\hfontseries{bc}\sffamily\selectfont
  \textcolor{TempColor}}
```

그림 4.4: 장절명령 스타일 정의

여기서 쓰인 색상(blue, DarkMagenta, TempColor)에 대해서는 아래 색상 관련 제 6 절에서 더 자세히 알아본다.

5.5 면주와 페이지스타일

면주(面注)는 흔히 PageStyle에서 "running heading"이라고 불리는 것이다. 이것을 정의하는 데는 fancyhdr 패키지가 쓰인다. 이 패키지만으로 원하는 거의 모든 설정을 할 수 있기 때문에 여기서 면주에 대하여 더 자세한 사항은 생략하려 한다. FAQ FancyHdr 페이지를 참고하라.

한 가지만 지적하자면, LaTeX은 "고아와 과부(orphans and widows)"라는 재미있는 이름으로 불리는 외따로 떨어진 줄을 허용하지 않는 훌륭한 기능이 있어서 예컨대 한 줄만이 외따로 한 페이지의 마지막이나 처음을 만들게 되면 그것을 다음 쪽으로 넘기는 경향이 있다. 이것은 조판상 올바르고 독자의 시선을 불필요하게 움직이지 않고 내용의 연결을 보장해준다는 점에서 반드시 적용되어야 할 문서 작성 규칙이지만, 워드 프로세서에 익숙한 일부 독자들은 앞 페이지의 아랫부분이 남거나 문단 사이가 조금 벌어지는 것을 한 줄, 심지어 절 제목 한 줄이 그 페이지의 마지막에 홀로 떨어지는 것보다 더 못참는 경향이 있는 듯하다.

이럴 경우 일부러 LaTeX의 규칙을 어기면서 "사회적으로 보호받아야 할 필요가 있는" 성분들을 다수 생산하는 것은 현명하지 못한 처사라 생각된다.[16]

다만, 앞 페이지를 종으로 채우고(vertically fill) 다음 페이지로 넘어갈 때 문단 간의 간격이 벌어지는 것을 못 참겠는 분을 위해서 `\raggedbottom` 명령 하나를 지적해두고 가겠다. 이 명령을 지정하면 페이지의 아래쪽 끝을 일부러 채우지 않는다. 그러므로 일부 문단이 다음 쪽으로 넘어가면 문단 사이의 간격을 벌리지 않고 페이지 아래를 비운다.

[16] 그래도 이 기능을 반드시 꺼야겠다는 분을 위해서, 이것은 TeX의 설정값 가운데 하나인 `\clubpenalty`와 `\widowpenalty` 값을 변경하여 동작을 제어할 수 있다는 점만 지적해두고자 한다. LaTeX은 이 값을 각각 150으로 정해두고 있다.

제 6 절 그림과 색상

6.1 그림 넣기

LaTeX에서의 그림 처리에 대해서는 훌륭한 문서들이 많다. KTUG의 GFaq도 그 중의 하나이다.

DVIPDFMx는 GhostScript를 이용하여 EPS 그림파일을 PDF 그림으로 변환하여 처리한다. 따라서 GhostScript가 잘 설치되어 있고 그림에 문제가 없다면 EPS 그림파일을 이용할 수 있다.

PDF나 JPG 또는 PNG 그림들을 그대로 이용할 수도 있다. 이 형식의 그림들이라면 ebb를 미리 실행하여 *.bb라는 확장명을 갖는 Bounding Box 파일을 미리 얻어두어야 한다는 점만 조심하면 될 것이다. 더 자세한 것은 여러 그림 처리 관련 문서를 참고하거나 graphicx 패키지 문서를 보라. 그림 처리는 기능 그 자체의 문제라기 보다는 만들어진 그림 자체의 품질이나 그림 처리의 "아이디어"가 더 큰 문제가 될 때가 많다.

6.2 색상

색상(color)은 color 패키지 또는 xcolor 패키지를 이용한다. xcolor는 강력한 색상 제어 기능을 제공하므로 관련 문서를 반드시 한 번 읽어보기 바란다.

이 문서에서는 RGB 모델을 이용하여 색상을 몇 개 설정해서 사용하고 있다. 인쇄목적이라면 CMYK 모델이 더 적합하겠지만 화면용이라면 RGB로도 충분할 것이다.

```
%% 색정의.
\definecolor{shadecolor}{rgb}{0.80,0.82,0.75}
\definecolor{DarkMagenta}{rgb}{0.75,0.25,0.25}
\definecolor{CautionBlack}{rgb}{0.10,0.10,0.10}
\definecolor{TempColor}{rgb}{0.10,0.80,0.20}
```

shadecolor는 framed 패키지에서 사용하는 기본값으로서 shaded 환경의 배경색으로 쓰인다.

제 7 절 그밖의 몇 가지 문제

7.1 PSTricks 문제

PDFLaTeX의 경우이든 DVIPDFMx의 경우이든 pstricks를 전혀 지원하지 못하는 것은 pstricks 패키지를 즐겨 사용하는 사용자의 입장에서는 안타까운 일이다. 그러나 이 문제는 PDF 번역기 자체의 문제로서 현실적으로 단기간에 해결될 가능성은 높지 않아 보인다. 현재까지 개발된 PDF에서 PSTricks 사용하기 방법은 대강 몇 가지가 있다. 예를 들면 pdftricks나 ps4pdf가 그런 것들이다. 그러나 이 해결책들은 모두 PDFLaTeX을 위한 것이다. 이 분야에서는 DVIPDFMx가 취약함을 인정하지 않을 수 없다.

GhostScript를 직접 이용하지 않는 한 대부분의 해결책은 pstricks로 그린 그림을 잘라내어서 EPS 그림으로 만들고 이것을 일반적인 그림과 마찬가지로 취급하여 불러들이는 방법이다. 이 방법들의 대강에 대해서는 FAQ PSTricks 페이지를 참고하라.

pstricks의 부수 스타일인 pst-eps 패키지는 TeXtoEPS 환경을 제공한다. 원하는 그림을 이 환경 안에 넣고 컴파일한 다음 dvips의 -E 옵션으로 변환하면 이 환경 안의 그림들을 EPS 파일들로 추출할 수 있다. 이 그림들을 원본 문서의 적절한 곳에 포함시킨다.

이런 식으로 해결책을 잘 생각해서 활용해보면 아마도 간단한 pstricks 그림들은 어렵지 않게 처리할 수 있지 않을까 한다.

7.2 기울인 글꼴 문제

이 문서에서는 한글 사체(斜體)를 전혀 사용하지 않았다. 영문에서의 강조 부분에 대해서 이탤릭 글꼴을 쓰는 것은 하나의 관행이지만, 한글 글꼴에서

는 이탤릭이란 있을 수 없으므로, 이탤릭이 주는 느낌과 유사하게 오른쪽으로 기울인 글꼴을 거기에 대응시킨 것은 HLaTeX의 선택이었다. 반면, 선택의 여지가 있기는 했지만 hLaTeXp의 경우에는 \emph나 \em 명령이 있는 곳에 그래픽 글꼴을 대응시켰다.

조판상의 이유로 후자의 방법이 나을 것으로 생각한다. \itshape가 불렸을 때 한글 글꼴을 사체에서 다른 서체로 바꾸어주려면 *.fd 폰트정의 파일을 수정하면 된다. 또는, 간단히 hitshape 패키지를 \usepackage하는 방법도 있다. 이 패키지는 한글에 대해서만 명조의 사체 대신 그래픽 글꼴을 사용하게 해준다.

7.3 밑줄 긋기

밑줄 긋기는 그다지 좋은 조판 관행은 아니다. 원래 이것은 원고를 타자기로 작성할 때 이탤릭체로 써야 할 곳을 표시하기 위해 쓰이던 도구였다. 그러나 이따금 밑줄을 그어야 할 때가 없는 것은 아닐 것이다.

\underline 명령은 비교적 잘 작동하지만 줄나눔이 되지 않는다. 밑줄 긋기에는 ulem 패키지가 흔히 쓰이는데, ulem이나 umoline이나 모두 한글 상황에서는 맞지 않아서 사용할 수 없었다.

myulem 패키지는[17] \uline 대신 \huline, \sout 대신 \hsout 등 h를 앞에 붙인 별도의 명령을 쓰도록 하고는 있지만 ulem 패키지의 기능을 대부분 제공한다. [normalem] 옵션도 작동한다. 다만, hlatex-interword 패키지와 함께 사용하는 경우, myulem 명령이 쓰인 범위 안에서는 자간 및 단어간격 설정이 작동하지 않는다.

[17] 다른 h-시리즈 패키지들과는 달리 이 패키지만 my--로 시작하는 이름을 가지고 있는데, 특별한 이유가 있는 것은 아니고 원래 이 패키지가 지극히 개인적인 용도로 작성되었기 때문이다. hulem으로 이름을 바꾸기엔 너무 늦어 있었다. :-)

표 4.1: 우리말 조사 규칙

앞단어 끝소리	와/과	을/를	이/가	은/는	(이)라	(으)로
리을(ㄹ)	과	을	이	은	이라	로
ㄹ 아닌 종성						으로
중성	와	를	가	는	라	로

7.4 상호참조 및 자동조사

LaTeX을 쓰는 즐거움 중의 하나가 상호참조와 인용을 자동으로 처리하는 것이 아닐까 한다. 그만큼 편리하고 좋은 기능이다. 한글 문서에서 상호참조 자체는 큰 문제가 없었다. 게다가 hLaTeXp와 HLaTeX은 상호참조로 불리워질 카운터의 호칭에 따라 자동으로 적절한 조사를 붙여주는 이른바 "자동조사"라는 탁월한 기능을 가지고 있다. 두 패키지 모두 hangul 패키지를 부르면 상호참조에 따르는 자동조사를 처리해주었다.

우리말의 조사 규칙은 표 4.1과 같다.

그런데, 이 자동조사 기능이 hyperref 패키지에서 문제를 일으켰다. 자동조사를 확인하기 위하여 삽입한 루틴이 hyperref의 label hyperlink 기능과 충돌한 것이다. 이와 관련된 내용은 34페이지의 4를 보라.

한참 동안 자동조사를 포기한 hangul-nojosa가 PDF 작성에 쓰였다. 이것은 엄청난 손실이었는데, 이제 hangul-k 패키지가 그 문제를 해결함으로써, 사실상 한글 PDF를 LaTeX으로 작성하는 것이 실질적으로 가능해졌다 할 것이다. 이로써 한글 PDF 문서를 어떤 희생도 없이 작성할 수 있는 길이 열린 것이다.

hangul-k에 관련된 자세한 내막은 부록 1을 보라.

앞 장에서 지적한 것이지만, hangul-k 패키지와 hyperref을 이용할 때 주의하여야 할 점은, label의 명칭은 반드시 영문으로 붙여야 한다는 점을 강조해둔다. 한글 label이나 인용 *key*에 대해서 hyperref이 에러를 내기 때문이다. 예컨대 `\label{그림1}`과 같은 한글 label은 허용하지 않는다. `\bibitem{아무개저자04}`와 같은 형식도 피하는 것이 좋다. hyperref

없이 한글 문서를 단지 hangul 패키지로만 작성하는 상황이라면 한글 label이나 한글 *cite_key*도 문제없다.

7.5 인용 숫자 압축

인용 숫자 압축이란, \cite 명령에 의해 만들어지는 인용의 참조가 숫자일 경우, 예컨대

```
\cite{bibref1, bibref2, bibref3, bibref4}
```

과 같은 명령에 의해 생성되는 연속되는 숫자들을 [1,2,3,4]와 같이 표시하지 않고 [1--4]와 같이 표시하는 기능을 말한다.

이 기능은 보통의 문서에서 cite 패키지에 의해 얻어졌다.

```
\usepackage{cite}
```

또는

```
\usepackage[sort&compress]{natbib}
```

한글 문서에서 hangul과 cite 패키지는 함께 사용되면 에러를 내지는 않지만 정렬·압축은 되지 않는다. 이 경우에는 hangul-nojosa를 쓴다.

```
\usepackage{cite}
\usepackage{hangul-nojosa}
```

그러나, hangul-nojosa는 자동조사를 포기해야 하기 때문에 별로 권장할 만하지 못하다. 그러므로, natbib 패키지를 쓰는 쪽이 낫다.

```
\usepackage{hangul}
\usepackage[sort&compress]{natbib}
```

그러나 위의 제시들은 PDF를 만들지 않는 상황에서의 해법이다. 문제는 hyperref 패키지를 사용할 때 발생한다.

hyperref 패키지는 cite 또는 natbib 패키지의 인용 숫자 압축 기능을 끈다. 왜냐하면 하이퍼링크를 만들는 경우 [1--4]는 중간의 2와 3의 \bibitem에

대해서 링크를 만들지 못하기 때문에 이것들을 차례로 열거해주어야 한다고 생각하기 때문이다. 하이퍼링크라는 관점에서는 이것이 올바른 방식일지 모르겠다. 그러나 인용 숫자 압축이 필요한 경우가 있으므로, hangul-k 패키지를 쓰는 상황에서 이 기능을 구현하려면 hypernat 패키지를 이용하여 다음과 같이 하여야 한다.

```
\usepackage{hangul-k}
\usepackage[dvipdfm,bookmarks=false]{hyperref}
\usepackage[sort&compress]{natbib}
\usepackage{hypernat}
```

이제 인용 숫자 압축을 얻을 수 있을 것이다. PDF 제작 상황에서 hangul-k 패키지를 사용한다면 cite 패키지는 적어도 인용 숫자 압축에 관한 한 사용하기 어렵다.

7.6 완성형 밖의 한글 쓰기

이제 한글 PDF 문서 작성에서 마지막 남은 문제는 이른바 "완성형 제한"의 문제였다. Omega(Ω)나 Lambda(Λ)를 사용하지 않고 LaTeX을 쓰는 한, "완성형"이라 알려져 있는 EUC-KR 범위의 문자밖에는 조판할 수 없다는 것은 중대한 한계였다. 이 문제를 근본적으로 해결하는 길은 TeX의 8비트 제한을 넘어서는 Omega(Ω)를 채택하는 도리밖에 없어 보인다.

윈도 운영체제가 CP949 한글을 채택함으로써 사실상 윈도상에서는 현대의 모든 한글을 다 표현할 수 있게 된 것과 비교할 때 HLaTeX이 시대에 뒤져보인 이유 가운데 하나가 이 "완성형 제한"이었을 것이다.[18]

KTUG에서는 완성형 표현의 범위를 넘어서는 한글 식자 방법에 대한 많은 토론과 기여가 이루어졌다. 그 결과, HLaTeX도 Lambda(Λ)를 이용하는 것이기는 하지만 CP949 한글을 식자할 수 있게 되었다. 그 후 조진환 님의

[18] 요즘의 윈도 사용자는 "완성형"이라 불리는 한글의 범위를 잘 알지 못할 것이다. KS X 1001은 한자, 상징문자를 모두 포함하고 있지만 참고자료로 한글에 해당하는 범위만 부록 4에 제시하여 두었다.

Omega-CJK, 김도현 님의 DHHangul과 같은 Omega(Ω)에 기반한 새로운 한글 문제 해결책들이 나오기 시작했다. hangul-k 패키지도 Lambda를 이용하여 손쉽게 CP949의 모든 현대 한글을 식자할 수 있다. 43페이지를 보라.

언젠가는 TEX에서의 한글 표현이 Omega(Ω)로 가게 될 것으로 믿는다.

LATEX의 한계 내에서 PDF 한글 구현을 문제삼는 경우, 완성형 이외의 문자를 식자할 수 있게 된다면 편리한 점이 있을 것이다. Lambda를 사용하지 못하는 상황도 있었기 때문이다. 특히 많은 완성형 외 문자를 사용하는 것이 아니라 한두 자의 중세문자나 CP949 문자를 이용해야 할 필요가 있을 때 그것 때문에 LATEX 문서를 Lambda 문서로 만드는 것은 쉬운 일이 아닐 수도 있는 것이다.

그래서 일종의 타협책으로 제시된 것이, CJK 패키지의 UTF8 환경을 이용하자는 것이었다. 이 환경은 유니코드의 UTF-8 인코딩된 텍스트를 처리해주는 기능이 있으므로, 이것을 이용해서 그다지 많지 않을 완성형 밖의 글자를 표현하는 데 쓰자는 발상이었다. 이래서 만들어진 것이 hlatexcjk 스타일 패키지이다. 이 패키지는 두 가지 명령과 한 개의 환경을 제공한다.

\urchr 명령 한 글자짜리 글자 파일을 만들어두고 이것을 불러들이는 명령이다. 예컨대 `ddom.char`라는 파일은 UTF-8 인코딩된 "똠" 글자의 유니코드 값을 가지고 있다. 이것을 `\urchr{ddom}`으로 불러와서 식자하는 것이다.

\UNI 명령 김도현 님이 제안하신 명령으로, 특정 글자의 16진수 유니코드 값을 인자로 취하여 그 글자를 찍어준다.

HCJK 환경 UTF-8 인코딩으로 작성된 외부 텍스트 파일을 불러들이는 환경이다.

유니코드 사용에서 가장 문제가 되는 것은 역시 글꼴이다. 은글꼴은 모든 현대 한글을 다 표현할 수 있지만 중세문자가 없다.[19]

중세문자 문제는 매우 복잡한데, 관심있는 분은 FAQ 옛한글처리 페이지를 방문해보기 바란다.

hlatexcjk를 쓰게 되는 이유 중에는 "흔글"과 같이 한두 글자의 고어를 사용해야 하는 경우가 많을 것이므로, 이럴 경우에 임시로 쓰게 하자는 것이 hlatexcjk의 제작목적이었기 때문에, 한컴바탕과 한컴돋움을 이용해서 그 글꼴에 내장된 완성형 중세문자를 그대로 가져다 쓰기로 하였다.[20]

실용적 목적으로 제작된 패키지이므로 꼭 사용할 필요가 있는 분만 사용해보시기 바란다. 다만, 크기가 만만찮은 HLaTeX과 CJK 패키지를 모두 불러들이다보니, 상당히 많은 메모리를 필요로 하는 점이 단점이다. 보통의 설정으로는 `\urchr` 명령으로 글자를 식자하는 데는 문제가 없겠지만 찾아보기를 만들면 메모리 부족을 호소할 지도 모른다.

[19] `UnBatang-odal.ttf`에는 중세문자 자소가 포함되어 있다. 그러나 일반적인 의미에서 고어 자소는 물론 완성형 고어를 포함하고 있지 않다고 해도 될 것이다.

[20] 이 방법은 일종의 편법이다. 중세문자 표현이 지향해야 할 바는 이렇게 글꼴 의존적으로 글꼴 자체의 사용자 영역에 심어져 있는 완성형 중세문자를 이용하는 방법이 아니라 "첫가끝" 방식이라 불리는 자소조합 형태로 구현되어야 온당하다고 생각한다.

제 5 장

마치는 말

TEX이든 LATEX이든 모두 글을 쓰기 위한 도구에 불과하다. 얼마나 훌륭한 문서를 만드느냐는 것은 도구의 성능에 달렸다기보다는 문서의 내용에 달린 것이 아닌가 한다. 단지 기술적으로 저자가 표현하고 싶은 것을 "표현할 수 있는 방법"이 있다는 것을 보여준 것으로 만족한다.

더 훌륭한 문서가 우리가 공들여서 마련한 다양한 패키지와 도구, 방법에 의하여 작성되고 공유되기를 희망한다. □

부록 A

부 록

A.1 hangul-k 패키지에 관하여

hangul-k는 hangul-nojosa를 대체하는 새로운 한글 문서 작성용 LaTeX third-party 패키지이다. 이 패키지는 김도현 님이 "하이퍼링크-자동조사" 해결책을 제시함으로써 만들어졌는데, "하이퍼링크-자동조사"란, 기존의 자동조사 해결방법과는 달리 hyperref 패키지가 하이퍼링크를 다 만들기를 기다려서 거기에서 반환되는 문자열의 마지막 한두 글자를 분석하여 알맞은 조사를 붙여주는 획기적인 방식이다.

이 작업은 2004년 4월 말에서 5월 초에 집중적으로 이루어졌다. 최초의 아이디어를 제시한 글은 4월 28–29일에 올라왔고, 이것을 필자가 시험해본 다음 hangul-nojosa와 합치는 일을 하였다. 그것이 5월 1일의 일이었다. 그 뒤에 전개된 일의 간단한 요약이다.

① 김도현 님 : 두 문자를 취하여 검사할 수 있음을 보임.
② 필자 : 두 문자 반환루틴의 버그 지적

③ 김도현 님 : 개선된 두 문자 검사 루틴 제안
④ 필자 : \if–\fi 형식의 한글 및 기호문자 검사 루틴 제안
⑤ 김도현 님 : josa.tab 파일을 읽어서 일반적으로 모든 EUC-KR 문자를 검사할 수 있도록 하는 루틴 제안
⑥ 필자 : "비참조 자동조사" 기능을 추가
⑦ 김도현 님 : hyperref 패키지와의 인터페이스 개선
⑧ 김도현 님 : 영문자 검사 루틴 개선, hyperref의 [pdftex] 옵션 지원 추가.
⑨ 필자 : josa.tab을 수정하고 \hRoman과 \hroman 명령 도입
⑩ 김도현 님 : '\이라' 명령 추가
⑪ 필자 : hyperref의 [dvips] 옵션 지원 추가.
⑫ 필자 : ksx1001-k.tex 호환 명령 추가. CP949 지원.

현재의 hangul-k 패키지는 대부분의 한글 문서를 멋지게 조판하는 것이 가능하고, 따라서 충분히 hangul 패키지를, 적어도 PDF를 작성하는 목적으로는, 대체할 수 있다고 생각한다. 단순히 자동조사가 추가된 것에 불과한 것이 아니라 여러 보충적인 명령과 기능을 추가할 수 있어서, 어떤 점에서는 hangul 패키지를 확장한 면도 있다 할 수 있다. 이것이야말로 지난 2년 넘게 고민해 온 문제가 해결된 셈이라 개인적으로는 정말 기쁜 일이었다.

A.2 hangul-k에서 DHHangul로

이 문서에서는 LaTeX으로 PDF 문서를 만드는 방법을 주로 다루었다. 최근의 DHHangul로 작업환경을 옮기려는 분이 있다면 이 글에서 다루어진 'h-시리즈 패키지'와의 호환성 여부가 궁금할 것으로 생각한다.

이 문제에 대해서 현재까지 알려진 내용을 요약해둔다.

참고로, DHHangul은 h-시리즈의 여러 설정들을 기본값으로 미리 채택해두고 있는 경우가 많다.

hlatexcjk 패키지. 이 패키지는 Omega 기반에서는 불필요하고, 사용하면 에러를 낸다. 따라서 hlatexcjk 관련 명령은 전부 없애고 유니코드 '문자'로 바꾸어 넣어야 한다.

hlatex-interword 패키지. 이 패키지 역시 HLaTeX에 고유한 명령을 재정의하고 있기 때문에 DHHangul에서 작동하지 않는다. DHHangul에서 자간은 `\CJKGlue`를 재정의하면 될 것이고, 단어간격은 표준적인 방식으로 `\spaceskip`을 설정하도록 한다. 예를 들면,

```
\spaceskip=.4em plus .1em minus .1em
```

hsetspace 패키지. DHHangul은 문서 행간 기본값으로 이미 hsetspace의 기본값을 채택하고 있다. 그래도 hsetspace를 사용하는 것이 불가능하지는 않을 것이다.

hsectsty 패키지. 별 문제 없이 사용가능할 듯하지만 sectsty로 충분하다. 그러므로 굳이 hsectsty를 쓰지 않아도 된다.

myulem 패키지. 사용할 수 없다. DHHangul에서 (아직) ulem 패키지가 원하는 대로 작동하지 않는다. 행이 바뀌는 밑줄을 그으려면 DHHangul에서는 umoline 패키지의 `\Underline`, `\Midline` 명령을 사용할 수 있다.

A.3 hangul-k 패키지 개정 연혁

현재 버전: 0.2h (2004/06/15)

2004-06-15 (0.2h) `\xjs`, `\fjs` 명령을 장절명령 안이나 `\bibitem`의 옵션 인자 문자열로 쓸 수 있게 개선. 이에 따라 사용자 설명서를 개정함.

2004-06-12 (0.2g) redundant `\expandafter` 제거 (suggested by DohyunKim)

2004-05-26 (0.2f1) 0.2f의 오류 제거.

2004-05-25 (0.2f) zip 압축파일로 제공함.
`ksx1001-k.tex`을 추가. Lambda가 실행되면 `ksx1001-k.tex`을 부르도록 함.

2004-05-18 (0.2e) HyperRef의 [dvips] 옵션 지원.

2004-05-12 (0.2d) 자동조사 명령 \이타 추가.
`\Hnum` 일부 수정.

2004-05-11 (0.2c) 일부 오류 수정. hyperlink 중복정의 삭제.
`\hroman`, `\hRoman` 명령 도입. `josa.tab` 수정.

2004-05-09 (0.2b) 영문자 처리루틴 개선 채택.(DohyunKim)

2004/05/04 (0.2a) hyperref의 `[dvipdfm]`과 `[pdftex]` 지원(DohyunKim)
hyperref 이후에 로드해야 하는 불편을 없앰.

2004/05/04 (0.2) DohyunKim의 일반화된 한글 기호문자 조사처리 방식의 채택. 새 버전으로 릴리즈.

A.4 EUC-KR 한글 문자 2350자

EUC-KR(KS X 1001) 한글 완성형 문자를 참고 자료로 제시한다. 한자와 상징 문자는 제외하였다. 한자와 상징 문자까지 포함하는 문자표는 http://examples.oreilly.com/cjkvinfo/AppL/ksx1001.pdf를 보라.

가각간갇갈갉갊감갑값갓갔강갖갗같 갚갛개객갠갤갬갭갯갰갱갸갹갼걀걋
걍걔걘걜거걱건걷걸걺검겁것겄겅겆 겉겊겋게겐겔겜겝겟겠겡겨격겪견겯
결겸겹겻겼경곁계곈곌곕곗고곡곤곧 골곪곬곯곰곱곳공곶과곽관괄괆괌괍
괏광괘괜괠괩괬괭괴괵괸괼괌괍괫굉 교굔굘굡굣구국군굳굴굵굶굻굼굽굿
궁궂궈궉권궐궜궝궤궷귀귁귄귈귐귑 귓규균귤그극근귿글긁금급긋긍긔기
긱긴긷길긺김깁깃깅깆깊까깍깎깐깔 깖깜깝깟깠깡깥깨깩깬깰깸깹깻깼깽
꺄꺅꺌꺼꺽꺾껀껄껌껍껏껐껑께껙껜 껨껫껭껴껸껼껫껬껭꼐꼬꼭꼰꼲꼴꼼
꼽꼿꽁꽂꽃꽈꽉꽐꽜꽝꽤꽥꽹꾀꾄꾈 꾐꾑꾕꾜꾸꾹꾼꿀꿇꿈꿉꿋꿍꿎꿔꿜

꿨꿩꿰꿱꿴꿸뀀뀁뀄뀌뀐뀔뀜뀝뀨끄 끅끈끊끌끎끓끔끕끗끙끝끼끽낀낄낌
낍낏낑나낙낚난낟날낡낢남납낫났낭 낮낯낱낳내낵낸낼냄냅냇냈냉냐냑냔
냘냠냥너넉넋넌널넓넘넙넛넜넝넣 네넥넨넬넴넵넷넸넹녀녁년널념녑녔
녕녘녜녠노녹논놀놂놈놉놋농높놓놔 놘놜놨뇌뇐뇔뇜뇝뇟뇨뇩뇬뇰뇹뇻뇽
누눅눈눋눌눔눕눗눙눠눴눼뉘뉜뉠뉨 뉩뉴뉵뉼늄늅늉느늑는늘늙늚늠늡늣
능늦늪늬늰늴니닉닌닐닒님닙닛닝닢 다닥닦단닫달닭닮닯닳담답닷닸당닺
닿대댁댄댈댐댑댓댔댕댜더덕덖던 덛덜덞덟덤덥덧덩덫덮데덱덴델뎀뎁
뎃뎄뎅뎌던덜뎠뎡뎨뎬도독돈돋돌돎 돐돔돕돗동돛돝돠돤돨돼됐되된될됨

됩됫됴두둑둔둘둠둡둣둥둬뒀뒈뒝뒤 뒨뒬뒵뒷뒹듀듄듈듐듑드득든듣들듦
듬듭듯등듸디딕딘딛딜딤딥딧딨딩딪 따딱딴딸땀땁땃땄땅땋때땍땐땔땜땝
땟땠땡떠떡떤떨떪떫떰떱떳떴떵떻떼 떽뗀뗄뗌뗍뗏뗐뗑뗘뗬또똑똔똘똥똬
똴뙈뙤뙨뚜뚝뚠뚤뚫뚬뚱뛔뛰뛴뛸뛤 뛥뛩뜨뜩뜬뜯뜰뜸뜹뜻띄띈띌띔띕띠
띤띨띰띱띳띵라락란랄람랍랏랐랑랒 랖랗래랙랜랠램랩랫랬랭랴략랸랏량
러럭런럴럼럽럿렀렁렇레렉렌렐렘렙 렛렝려력련렬렴렵렷렸령례롄롑롓로
록론롤롬롭롯롱롸롼뢍뢨뢰뢴뢸룀룁 룃룅료룐룔룝룟룡루룩룬룰룸룹룻룽
뤄뤘뤠뤼뤽뤤뤨뤴뤔뤗류륙륜률륨륩 륫륭르륵른를름릅릇릉릊릍릎리릭린

릴림립릿링마막만많맏말맑맒맘맙맛 망맞맡맣매맥맨맬맴맵맷맸맹맺먀먁
먈먕머먹먼멀멂멈멉멋멍멎멓메멕멘 멜멤멥멧멨멩며멱면멸몃몄명몇메모
목몫몬몰몲몸몹못몽뫄뫈뫘뫙뫼묀묄 묍묏묑묘묜묠묩묫무묵묶문묻물묽묾
뭄뭅뭇뭉뭍뭏뭐뭔뭘뭡뭣뭬뮈뮌뮐뮤 뮨뮬뮴뮷므믄믈믐믓미믹민믿밀밂밈
밉밋밌밍및밑바박밖밗반받발밝밞밟 밤밥밧방밭배백밴밸뱀뱁뱃뱄뱅뱉뱌
뱍뱐뱝버벅번벋벌벎범법벗벙벚베벡 벤벧벨벰벱벳벴벵벼벽변별볍볏볐병
볕볘볜보복볶본볼봄봅봇봉봐봔봤봬 뵀뵈뵉뵌뵐뵘뵙뵤뵨부북분붇불붉붊
붐붑붓붕붙붚붜붤붰붸뷔뷕뷘뷜뷩뷰 뷴뷸뷤뷧뷩브븍븐블븜븝븟비빅빈빌

빎빔빕빗빙빚빛빠빡빤빨빪빰빱빳빴 빵빻빼빽뺀뺄뺌뺍뺏뺐뺑뺘뺙뺨뻐뻑
뻔뻗뻘뻠뻣뻤뻥뻬뼁뼈뼉뼘뼙뼛뼜뼝 뽀뽁뽄뽈뽐뽑뽕뾔뾰뿅뿌뿍뿐뿔뿜뿟
뿡쀼쀵쁘쁜쁠쁨쁩삐삑삔삘삠삡삣빵 사삭삮산삳살삵삶삼삽삿샀상샅새색
샌샐샘샙샛샜생샤샥샨샬샴샵샷샹섀 섄섈섐섕서석섞섟선섣설섦섧섬섭섯
섰성섶세섹센셀셈셉셋셌셍셔셕션셜 셤셥셧셨셩셰센셸셺소속솎손솔솖솜
솝솟송솥솨솩솬솰솽쇄쇈쇌쇔쇗쇘쇠 쇤쇨쇰쇱쇳쇼쇽숀숄숌숍숏숑수숙순
숟술숨숩숫숭숯숱숲숴쉈쉐쉑쉔쉘쉠 쉥쉬쉭쉰쉴쉼쉽쉿슁슈슉슐슘슛슝스
슥슨슬슭슴습슷승시식신싣실싫심십 싯싱싶싸싹싻싼쌀쌈쌉쌌쌍쌓쌔쌕쌘

쌜쌤쌥쌨쌩썅써썩썬썰썲썸썹썼썽쎄 쎈쎌쎈쏘쏙쏜쏟쏠쏢쏨쏩쏭쏴쏵쏸쐈
쐐쐤쐬쐰쐴쐼쑈쑤쑥쑨쑬쑴쑵쑹쒀 쒔쒜쒸쒼쓰쓱쓴쓸씀씈씀씁씌썬쓀
씜씨씩씬씰씸씹씻씽아악안앉않알앍 앎앓암압앗았앙앝앞애액앤앨앰앱앳
앴앵야약얀얄얇얌얍얏양얕얗얘얜얠 얩어억언얹얻얼얽얾엄업없엇었엉엊
엌엎에엑엔엘엠엡엣엥여역엮연열엶 엷염엽엾엿였영옅옆옇예옌옐옘옙옛
옜오옥온올옭옮옰옳옴옵옷옹옻와왁 완왈왐왑왓왔왕왜왝왠왬왯왱외왹왼
욀욈욉욋욍요욕욘욜욤욥욧용우욱운 울욹욺움웁웃웅워웍원월웜웝웠웡웨
웩웬웰웸웹윈위윅윈윌윔윕윗윙유육 윤율윰윱윳융윷으윽은을읊음읍읏응

읒읓읔읕읖읗의읜일읨읫이익인일읽 읾잃임입잇있잉잊잎자작잔잖잗잘잚
잠잡잣잤장잦재잭잰잴잼잽잿쟀쟁쟈 쟉쟌쟎쟐쟘쟝쟤쟨쟬저적전절젊점접
젓정젖제젝젠젤젬젭젯젱져젼젤졈졉 졌졍졔조족존졸졺좀좁좃종좇좋좡좌
좍좔좝좟좡좨좼좽죄죈죌죔죕죗죙죠 죡죤죵주죽준줄줅줆줌줍줏중줘줬줴
쥐쥑쥔쥘쥠쥡쥣쥬쥰쥴쥼즈즉즌즐즘 즙즛증지직진짇질짊짐집짓징짖짙짚
짜짝짠짢짤짧짬짭짯짰짱째짹짼쨀쨈 쨉쨋쨌쨍쨔쨘쨩쩌쩍쩐쩔쩜쩝쩟쩠쩡
쩨쩽쪄쪘쪼쪽쫀쫄쫌쫍쫏쫑쫓쫘쫙쫠 쫬쬐쬈쬐쬔쬘쬠쬡쭈쭉쭌쭐쭘쭙쭝

쮜쭸쭹쮜쮸쯔쯤쯧쯩찌찍찐찔찜찝찡 찢찧차착찬찮찰참찹찻찼창찾채책챈

챌챔챕챗챘챙챠챤챦챨챰챵처척천철 첨첩첫첬청체첵첸첼첨첩첫첬청체첵첸첼첨첩첫첬청처쳔쳤
쳬쳰쳉초촉촌촐촘촙촛총촤촨촬쵱최 쵠쵤쵬쵭쵯쵱쵸쵬추축춘출춤춥춧충
춰췄췌췐취췬췰췸췹췻췽츄츈츌츔츙 츠측츤츨츰츱츳층치칙친칟칠칡침칩
칫칭카칵칸칼캄캅캇캉캐캑캔캘캠캡 캣캤캥캬캭컁커컥컨컫컬컴컵컷컸컹
케켁켄켈켐켑켓켕켜켠켤켬켭켯켰켱 켸코콕콘콜콤콥콧콩콰콱콴콸콤콥콧콩콰콱콴콸콤콩쾌
쾡쾨쾰쿄쿠쿡쿤쿨쿰쿱쿳쿵쿼퀵퀀퀄퀑 퀘퀭퀴퀵퀸퀼큄큅큇큉큐큔큘큠크큭
큰클큼큽킁키킥킨킬킴킵킷킹타탁탄 탈탉탐탑탓탔탕태택탠탤탬탭탯탰탱
탸턍터턱턴털턺텀텁텃텄텅테텍텐텔 템텝텟텡텨텬텼톄톈토톡톤톨톰톱톳

통톺톼톈퇘퇸퇫툉툐투툭툰툴툼툽 툿퉁퉈퉜퉤튀튁튄튈튐튑튕튜튠튤튬
튱트특튼튿틀틂틈틉틋틔틘틸틤팁팃 틱틴틸팀팁팃팅파팍팎판팔팖팜팝팟
팠팡팥패팩팬팰팸팹팻팼팽퍄퍅퍼퍽 펀펄펌펍펏펐펑페펙펜펠펨펩펫펭펴
편펼폄폅폈평폐폘폡폣포폭폰폴폼폽 폿퐁퐈퐝푀푄표푠푤푭푯푸푹푼푿풀
풂품풉풋풍풔풩퓌퓐퓔퓜퓟퓨퓬퓰퓸 퓻퓽프픈플픔픕픗피픽핀필핌핍핏핑
하학한할핥함합핫항해핵핸핼햄햅햇 했헹햐향허헉헌헐헒험협혓헝혜헥헨
헬헴헵헷헹혀혁현혈혐협혓혔형혜혠 혤혭호혹혼홀홅홈홉홋홍홑화확환활
홧황홰홱홴횃횅회획횐횔횝횟횡효횬 횰횹횻후훅훈훌훑훔훗훙훠훤훨훰훵

훼훽훤휄휑휘획휜휠휨휩휫휭휴휵휸 휼흄흇흉흐흑흔흖흗흘흙흠흡흣흥흩
희흰흴흼흽힁히힉힌힐힘힙힛힝

A.5 이 문서의 Preamble

이 문서의 Preamble 설정을 예제로 제시한다.

```
%% Preamble of hangul-k-manual.tex
%% PDF 크기설정 \special 명령
\AtBeginDvi{\special{pdf: pagesize width 150mm height 212mm}}
%% hlatexcjk package requires CJK and hangul packages.
\usepackage{CJK}
\usepackage{hangul-k}

%% PaperSize Setting.
\usepackage[paperwidth=150mm,
    paperheight=212mm,
    hmargin=20mm,
    top=25mm,
    bottom=27mm]{geometry}
\setlength\headheight{27.5pt}

%% Hyperlink and bookmarks for PDF output.
%% Ask to user if Hangul PDF bookmark is to be made.
\usepackage{ifthen}
\makeatletter
 \chardef\ttyin16
 \chardef\ttyout16
 \def\iden#1{#1}
 \def\strip#1#2 \@gobble{\def #1{#2}}
 \def\@defpar{\par}
 \def\yes{yes}
 \def\y{y}
 \def\n{n}
 \def\Msg{\immediate\write\ttyout}
 \def\Ask#1#2{%
    \message{#2}\read\ttyin to #1\ifx#1\@defpar\def#1{}\else
      \iden{\expandafter\strip
 \expandafter#1#1\@gobble\@gobble} \@gobble\fi}
```

```
\makeatother

\newboolean{YesOrNo}
\setboolean{YesOrNo}{true}

\def\getYesOrNo{%
\Msg{^^J%
 *************************************************************^^J%
 * Do you want to make HANGUL PDF bookmarks, Yes or No(y/n)?}%
\Ask\answer{%
 *************************************************************}%
}

\newcommand\DetermineYesOrNo{%
%% 만약 \answer가 이미 주어져있지 않다면
 \ifx\answer\undefined
         \getYesOrNo
 \else\fi
%% \answer에 따라 YesOrNo 불린값을 결정함
 \ifx\y\answer\setboolean{YesOrNo}{true}
         \else\setboolean{YesOrNo}{false}\fi
}

\DetermineYesOrNo

\ifthenelse{\boolean{YesOrNo}}
 {% true
  \usepackage[dvipdfm,CJKbookmarks,colorlinks]{hyperref}
  \AtBeginDvi{\special{pdf: tounicode KSCms-UHC-UCS2}}
 }
 {% false
  \usepackage[dvipdfm,bookmarks=false]{hyperref}
 }
\hypersetup{%
     pdftitle={hangul-k 사용설명서},
     pdfauthor={김강수},
     pdfsubject={HLaTeX-0.991 document style package, hangul-k},
```

```
     pdfkeywords={hangul, hangul-k, PDF, HLaTeX, manual, KTUG},
}

%% for DVIPDFMx pre-20040320.
\makeatletter
\AfterBeginDocument{%
  \ifx\special@paper\@empty\else
    \ifHy@setpagesize
      \special{papersize=\special@paper}%
    \fi
    \Hy@DisableOption{setpagesize}%
  \fi
}
\makeatother

%% 한글 이름 설정.
\ksnamedef{contentsname}{차~례}
\ksnamedef{bibname}{참고~문헌}
\ksnamedef{listfigurename}{그림~차례}
\ksnamedef{listtablename}{표~차례}
\ksnamedef{glossaryname}{용어집}

%% verb, mflogo, hlatexcjk.
\usepackage{fancyvrb} % before hlatexcjk
\usepackage{mflogo} % METAFONT logo.
\usepackage{hlatexcjk}

%% Now, define the layouts of the document.
%% CM and UNTTF.
\usepackage{unttf}
\let\textpl\textpg
\let\plfamily\pgfamily
\let\textar\textbm

%% 우리말 명령 두 개.
%% 예컨대 '\바탕'은 사실은 '\^^b9'일 뿐이므로 다른
%% '\^^b9' 명령이 있을 경우에는 주의해서 정의하여야 한다.
```

```
\def\바탕{\hfontfamily{bt}\ignorespaces}%  바탕  (B9D9C5C1)
\def\돋움{\고딕}%

\usepackage[default]{hlatex-interword} % 자간, 단어간격
\usepackage{hitshape} % 한글 \itshape의 그래픽글꼴 처리.
\usepackage[normalem]{myulem} % 한글 밑줄.
%\usepackage[ensec]{hsectsty}
\usepackage{hsectsty} % 한글 장절명령 스타일.
\sectionfont{\noindent\hfontseries{bc}\textcolor{blue}}
\subsectionfont{\noindent\hfontseries{bc}\textcolor{DarkMagenta}}
\subsubsectionfont{\noindent\hfontseries{bc}\sffamily\selectfont
  \textcolor{TempColor}}
\paragraphfont{%
  \noindent\hfontseries{bc}\selectfont\textcolor{ParagraphColor}}

%% Graphics, Colors, Frames
\usepackage{graphicx,color}
\usepackage{xcolor}
%% 색정의.
\definecolor{shadecolor}{rgb}{0.80,0.82,0.75}
\definecolor{DarkMagenta}{rgb}{0.75,0.25,0.25}
\definecolor{TempColor}{rgb}{0.10,0.80,0.20}
\definecolor{ParagraphColor}{rgb}{0.30,0.30,0.60}
\usepackage{framed} % framed, shaded 환경.
\usepackage{boxedminipage} % boxedminipage 환경.

%% MakeIndex
\usepackage{makeidx}

\usepackage{url}

%% For Tabular
\usepackage{multirow,array,mdwtab}

%% list, compactlist
\usepackage{mdwlist}
\makecompactlist{enum**}{enumerate}
```

```
%% listing and verbinput.
\usepackage{sverb}

%% floats.
\renewcommand{\topfraction}{.85}
\renewcommand{\bottomfraction}{.5}
\renewcommand{\textfraction}{.15}
\renewcommand{\floatpagefraction}{.66}

%% Page headings
\usepackage{calc}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead{}
\fancyhead[LO,LE]{\includegraphics[width=14mm]{ktugimage}}
\fancyhead[RO]{\colorbox{shadecolor}%
    {\minipage{\linewidth - 17mm}\raggedleft
     \sffamily\small\rightmark\qquad \thepage\endminipage}}
\fancyhead[RE]{\colorbox{shadecolor}%
    {\minipage{\linewidth - 17mm}\raggedleft
     \sffamily\small\leftmark\qquad \thepage\endminipage}}
\chead{}\cfoot{}\rfoot{}\lfoot{}
\renewcommand\chaptermark[1]{\markboth{#1}{}}
\renewcommand\sectionmark[1]{%
    \markright{제\,\thesection\,절\quad #1}}

%% LayOut Display
\usepackage{layouts}

%% Footnotes.
\usepackage[bottom]{footmisc}
\raggedbottom
%\usepackage{preview}

%% Example Environment. by Karnes.
\usepackage{lsexa}
```

```
%% Author's commands and environments.
\newcommand\cntrdot{%
  \ifvmode\leavevmode\fi%
  $\,\cdot\,$%
}

\newcommand\bnm[1]{%
  \ifvmode\leavevmode\else\fi
%  \kern -.45em『#1』\kern -.5em%
%% 위의 설정은 한컴바탕 글꼴용. 은글꼴은 커닝이 불필요하다.
%% 글꼴마다 이 부호의 커닝값이 다름.
 『#1』%
}
%% bookname 명령.

\newcommand\snm[1]{%
  \ifvmode\leavevmode\else\fi
%  \kern -.45em「#1」\kern -.5em%
%% 위의 설정은 한컴바탕 글꼴용. 은글꼴은 커닝이 불필요하다.
%% 글꼴마다 이 부호의 커닝값이 다름.
 「#1」%
}

\newcommand\cmd[1]{%
  \index{명령!\textbackslash #1}\texttt{\textbackslash #1}%
}

\newcommand\pkg[1]{%
  \index{패키지!#1}\textsf{#1}%
}

\newcommand\cls[1]{%
  \index{클래스!#1}\textsf{#1}%
}

\newcommand\file[1]{%
```

```
  \index{파일!#1}\texttt{#1}%
}

\newcommand\prgrm[1]{%
  \index{프로그램!#1}\textsf{#1}%
}

\newcommand\env[1]{%
  \index{환경!#1}\texttt{#1}%
}

\newcommand\teximp[1]{%
  \index{TeX@\THETeX!#1}\textrm{#1}%
}

\newcommand\THETeX{%
    \protect\TeX%
}
%% 이 이상한 명령을 두게 된 이유는 색인 만들기 때문.
%% HyperRef을 얹으면 색인 명령에서 \TeX이 풀려버린다.

\newcommand\wi[1]{%
  #1\index{#1}%
}

\newcommand\hangulk{%
    \pkg{hangul-k} \DEFINEJOSA{패키지}%
}

\newcommand\MiKTeX{%
    \index{TeX@\THETeX!MiK\THETeX}MiK\THETeX%
}

\renewcommand\HLaTeX{%
    H\LaTeX%
}
```

```
%% original \HLaTeX logo.
\newcommand\HHLaTeX{%
        한\kern-.4ex\lower.3ex\hbox{글}\kern-.4ex\LaTeX}%

\newcommand\NFSS{%
    \index{글꼴!NFSS}\textsf{NFSS}%
}

\newcommand\HFSS{%
    \index{글꼴!HFSS}\textsf{HFSS}%
}

\hyphenation{quo-ta-tion}

%% index 환경의 재정의
\makeatletter
\renewenvironment{theindex}
    {\if@twocolumn
        \@restonecolfalse
      \else
        \@restonecoltrue
      \fi
      \columnseprule \z@
      \columnsep 35\p@
      \twocolumn[\section*{\indexname}]%
      \@mkboth{\MakeUppercase\indexname}%
              {\MakeUppercase\indexname}%
      \thispagestyle{fancy}%
      \phantomsection
      \addcontentsline{toc}{chapter}{\protect\indexname}%
      \parindent\z@
      \parskip\z@ \@plus .3\p@\relax
      \let\item\@idxitem}
     {\if@restonecol\onecolumn\else\clearpage\fi}
%% glossary 환경의 재정의. 기본적으로 index 환경과 같게.
\renewenvironment{theglossary}
    {\if@twocolumn
```

```
        \@restonecolfalse
      \else
        \@restonecoltrue
      \fi
      \columnseprule \z@
      \columnsep 35\p@
      \twocolumn[\section*{\glossaryname}]%
      \@mkboth{\MakeUppercase\glossaryname}%
              {\MakeUppercase\glossaryname}%
      \thispagestyle{fancy}%
      \phantomsection
      \addcontentsline{toc}{chapter}{\protect\glossaryname}%
      \parindent\z@
      \parskip\z@ \@plus .3\p@\relax
      \let\item\@idxitem}
     {\if@restonecol\onecolumn\else\clearpage\fi}
\makeatother

%% maketitle
%% got from http://zoonek.free.fr/LaTeX/LaTeX_samples_title/0.html
\makeatletter
\def\thickhrulefill{%
    \leavevmode \leaders \hrule height 1pt\hfill \kern \z@}
\renewcommand{\maketitle}{\begin{titlepage}%
    \pagecolor{shadecolor}%
    \let\footnotesize\small
    \let\footnoterule\relax
    \parindent \z@
    \reset@font
    \null
    \vskip -5\p@
    \hbox{\mbox{%
        \hspace{-4pt}%
        \fbox{\includegraphics[width=3em]{ktugimage}}%
        \hspace{4pt}
        }%
      \vrule depth 0.9\textheight%
```

```
      \mbox{\hspace{2em}}
      \vtop{% %%%%%%%%%%%%%%%%%%
        \vskip 40\p@
        \begin{flushleft}
          \huline{\Large \textgr{\@author}}\par
        \end{flushleft}
        \vskip 80\p@
        \begin{flushleft}
%% remove next line to make final version.
%          \Large\itshape DRAFT \\
          \huge \bfseries \@title \par
        \end{flushleft}
        \vfil
        }}
    \null
  \end{titlepage}%
  \pagecolor{white}%
  \setcounter{footnote}{0}%
}
\makeatother

%%% 92% 장평 한글 문자 사용을 위한 추가 정의.
%% Caption thinhangul setting.
\usepackage{ccaption}
\captiontitlefont{\normalfont\hfontseries{c}}
\captionnamefont{\normalfont\hfontseries{c}}

%% description label thinhangul setting.
\renewcommand\descriptionlabel[1]{%
    \hspace\labelsep
    \normalfont\hfontseries{c}\bfseries #1}

%% footnote thinhangul setting.
\let\origfootnotesize\footnotesize\relax
\def\footnotesize{\origfootnotesize\hfontseries{c}}

%% Parindent.
```

```
\setlength\parindent{1em}

%% eso-pic. cover.eps.
\usepackage{eso-pic}
\newcommand\CoverPicture{%
    \parbox[b][\paperheight]{\paperwidth}{%
        \vfill
        \centering
        \includegraphics[width=150mm,keepaspectratio]%
            {cover}%
%       \vfill
}}

%% Index
\makeindex \makeglossary

%% Chapter. hfncychap
\usepackage[Glenn]{hfncychap}
\ChNameVar{\raggedleft\normalsize\rm\hfontfamily{mg}}
\ChNumVar{\raggedleft\Huge\hfontfamily{mg}}
\ChTitleVar{\raggedleft\Huge\rm\hfontfamily{mg}}

%% crop. for draft.
%\usepackage[a4,cross]{crop}

%% 행간
\usepackage[hangul,adjustfootnotesep,adjustverbatim]{hsetspace}
\SetHangulspace{1.333}{1.15}
\renewcommand\footnotesep{8pt}

%% 백면.
%% from Lee Juho.
\makeatletter
\newcommand{\cedp}{%
    \if@openright
    \newpage{%
        \pagestyle{empty}
```

```
        \ClearShipoutPicture
        \cleardoublepage}%
    \else
    \newpage{%
        \pagestyle{empty}
        \ClearShipoutPicture
        \clearpage}%
    \fi
}
\makeatother

%% Title page settings.
\title{hangul-k 사용 설명서}
\author{도은이아빠}
\date{\today}

%% End of preamble. Now, start the document.
% \endinput
```

참고문헌

[고기형1999] 고기형 외, 「Unicode 한글 TEX 개발」, 연구보고서, 문화관광부. 1999.

[은광희2000] 은광희, 『HLATEX 길잡이』, 0.991 판, 2000년 1월 18일, 온라인문서. http://faq.ktug.or.kr/mywiki/_c7_d1_b1_dbLaTeX_b1_e6_c0_e2_c0_cc.

[차재춘1998] 차재춘, 『한글 HLATEXp 사용자 설명서』, 1998년 11월 20일, 온라인문서.

[Syropoulos2003] A. Syropoulos, *Digital Typography Using LATEX*, Springer-Verlag. 2003.

찾아보기

용어집

감사의 말

이 길지 않은 글을 쓰는 데도 여러 사람의 도움을 입었다. 여기에 감사의 뜻을 밝혀 적어둔다.

HLaTeX을 만들고 UHC 글꼴을 공개하신 은광희 님께 감사드린다. KTUG에서 한글 사용에 관한 모든 노력은 HLaTeX에서부터 출발한 것이었다. UHC 글꼴의 공개는 그 글꼴을 만들기 위해 들여야 할 노력을 생각하면 특별히 감사의 말씀을 드리지 않을 수 없다. 또한, 이 매뉴얼의 중요한 부분 자체가 은광희 님의 「한글LaTeX 길잡이」의 인용으로 이루어져 있다.

DVIPDFMx와 MiKTeX-KTUG을 만들고 유지해오신 조진환 님께 감사드린다. 일일이 적을 수 없을 정도로 한글 LaTeX 사용에 있어서는 결정적인 기여들을 거의 혼자서 해내신 데 대해서 감사와 존경의 뜻을 적어둔다. DVIPDFMx가 없었더라면 이 모든 작업이 무의미했을 것이다.

hangul-k 패키지의 사실상의 저자이신 김도현 교수께 감사드린다. 예전 위키 홈페이지에 "사회과학도가 TeX을 알면 외로워진다"고 적으셨던 견해가 바뀌셨기를 바라 마지 않는다. 김도현 교수의 열린 노력이 한글 LaTeX

사용자들에게 끼친 바는 매우 크다.

은글꼴을 만들고 공개하신 박원규 님께 감사드린다. KTUG은 알게 모르게 박원규 님께 빚지고 있는 것이 많다.

글쓰는 과정에 관심을 갖고 개인적으로나 KTUG을 위해서나 물심양면의 지원을 아끼지 않으시는 남상호 박사께 감사드린다.

초고를 읽고 칭찬과 적절한 지적을 보여주신 이호재 님, 이기황 님께 감사드린다.

이주호 님은 사실상 KTUG 문서화 팀의 중심이다. 이 글도 이주호 님의 격려와 지원이 없었으면 불가능했을 것이다. 이 글을 쓰면서 참고한 몇몇 문서는 이주호 님의 배려로 열람할 수 있었다.

이름을 일일이 열거하지 못한 많은 분들, 그리고 이 글을 읽어주실 독자들께도 미리 감사드린다.

이 글의 모든 잘못과 부족한 점은 필자의 것이다. 이 문서에서 조금이라도 쓸 만한 부분이 있다면 모두 위에 든 분들께서 지적해주시거나 가르쳐주신 내용 중에서 나온 것이다.

2004년 6월 19일

김 강 수 씀.