한글 글꼴 설치 사용법 — Hangul-ucs를 중심으로

노바디

2006년 4월 29일

요 약

hangul-ucs 패키지가 제공하는 기본 글꼴은 은글꼴 트루타입이다. 기본 글꼴만으로도 훌륭한 문서를 물론 작성할 수 있을 것이다. 그러나 사용자가 자신의 새로운 폰트 세트를 조성하거나 다른 사람이 만든 폰트 패키지를 설치하여 사용하려면 약간의 추가적인 지식이 필요하다. 이 글에서는 문화부 type 1 글꼴을 설치하여 문서에 활용하는 방법을 소개한다.

차 례

1 글머리에

엊그제 조인성 교수와 대화 중에 “폰트 문제”에 대한 언급이 있었다. 일반 사용자 입장에서 새로운 폰트를 설치해서 쓰는 것이 쉽지 않다는 것이 주된 말씀이셨는데, 이 점에 대해서 쉬운 가이드를 제시하지 못한 점을 깊이 반성하고, 문화부 type1 글꼴을 설치하여 본문에서 활용하는 방법에 대한 간단한 가이드를 하나 쓰기로 한다.

이 글의 주안점은 다음과 같다.

(1) 쉬울 것. 두 단계 이상 넘어가면 골치아파하는 사용자를 위해서 쓰는 글이어야 한다.

(2) 활용성이 높을 것. 이 글에서 익힌 방법을 다른 경우에도 응용할 수 있도록 해야 한다.

쉽지 아니한 요구임에 틀림없으나, 한번 시도해보자.

2 한글 글꼴의 설치 사용을 위해 필요한 기본 지식

2.1 영문 글꼴, 한글 글꼴

> 한글 영역과 영문자 영역에 서로 다른 글꼴을 사용한다.

hangul-ucs는 문서에서 사용되는 폰트를 모두 네 영역으로 나눈다.

1. 영문 텍스트 글꼴
2. 수학 글꼴
3. 한글 영역 글꼴
4. 한자 및 기호문자 영역 글꼴

이 가운데 1과 2는 영문 글꼴 선택 방식과 동일하고 3과 4는 일괄 설정하게 되어 있으므로 결과적으로 “영문 글꼴”과 “한글 글꼴”의 두 부류로 나누어 처리하고 있다고 말할 수 있다. 이 “글꼴 분리 정책”이 나오게 된 사연은 다음과 같다. 원래 hLaTeXp나 HLaTeX은 모두 한글 글꼴 문제에서 상당한 고생을 하면서 작성된 패키지였다. hLaTeXp의 경우 제한된 라이센스 글꼴을 사용해야 했던 문제가 있었으며, HLaTeX의 경우는 거의 모든 글꼴을 처음부터 디자인해야 하는 어려움을 겪었다. 이 당시 한글 LaTeX 패키지의 디자인 개념은 우선 “한글을 표현 가능하게 한다”는 데 주력하고 있었던 까닭에 영문 글꼴 영역은 영문의 CM 글꼴을 쓰는 것을 당연시하였다. 예컨대 hLaTeXp의 경우

영문 글꼴은 무조건 CM 하나로만 통일시켜 두었던 것이다. 그러다가, HLaTeX이 Lambda 기반으로 이행할 준비를 하면서 한글 글꼴의 영문자 영역을 영문 식자에 활용하는 쉬운[1] 해결책을 채택함으로써 HLaTeX-Lambda에서 단일 글꼴을 사용하는 새로운 시도가 이루어졌다. 그러나 생각건대 단일 글꼴 접근이라는 코드가 간명하고 개념상으로도 바람직한 접근 방법이 실효를 거두려면 글꼴 자체의 품위가 충분히 준비되어 있어야 한다. 즉, 최소한 Cyberbit 정도의 자소를 갖춘 유니코드 거의 전 영역 문자를 포함하는 양질의 폰트가 필요하다는 것이다. 애석하게도 그러한 글꼴을 갖추지 못하였다. UHC의 명조(mj) 글꼴은 은광희 님의 초인적인 노력에 의해 얼마간 한글 문서 식자에는 어려움이 없었으나, 동유럽 문자를 비롯한 라틴 계열, 그리고 그밖의 다국어 문자는 손도 대지 못하였고, 한자 영역도 EUC-KR 영역만을 카버하는 정도로 그쳤기 때문에, 단일 글꼴 접근은 오히려 문서 작성을 번거롭게 만드는 결과를 가져왔다.

dhhangul이 개발되던 과정에서 나는 "글꼴 분리 정책"을 강력하게 요구했다. 왜냐하면 영문자 글꼴 영역을 분리해두어야 다양한 영문 글꼴을 활용할 수 있고 복잡한 라틴 문자를 `babel` 패키지를 이용하여 식자할 수 있을 것이라 생각하였기 때문이다. 이것이 dhhangul의 가장 중요한 특징 가운데 하나가 되었다. 그리고 이 정책이 hangul-ucs까지 이어진 것이다. 실제 한글 글꼴의 영문자 영역을 영문 식자에 사용해본 테스트 결과에 의하면 그다지 고무적인 결과를 얻지 못하였다. 한글 폰트의 영문자 영역은 영어 50여자 정도만이 제대로 마련되어 있을 뿐 문장부호나 기타 문자에 대해서 디자인上의 고려가 부족하다는 느낌을 받았다.

그러므로, hangul-ucs 문서에서 영문 영역 글꼴의 선택은 일반적인 라틴 문자 식자 방법을 그대로 이용한다.

2.2 NFSS, LaTeX 2ε의 폰트 선택 체계

> LaTeX에서 글꼴을 다루려면 NFSS (New Font Selection Scheme)에 대한 **기본적인** 이해가 꼭 필요하다.

의식하고 있지 않더라도 `\textbf`와 같은 명령을 쓰고 있다면 이미 NFSS를 사용하고 있는 셈이다. 즉 LaTeX 2ε를 사용하고 있다면 NFSS라는 LaTeX 2ε의 글

1) Lambda에서는 글꼴 영역 분리가 훨씬 어렵다.

꼴 선택 방식에 당연히 친숙하다고 할 수 있다. NFSS에 여러 주제가 있지만 여기서는 텍스트 폰트에 대해서만 알아본다. 이미 언급한 대로 수학 글꼴의 설정과 사용은 이 글의 주제가 아니기 때문이다. 이 절의 내용은 내가 KTUG Faq에 적은 글[2]에서 대부분 가져왔다.

2.2.1 텍스트 폰트의 속성

모든 텍스트 폰트는 다섯 가지 속성(attributes)을 가진다.

인코딩(encoding) 폰트 인코딩이란 폰트 속에 글자들이 나열되어 있는 순서를 가리킨다. TEX-류에서 흔히 쓰이는 폰트 인코딩은 두 가지인데 하나는 D. Knuth가 만든 TEX text encoding(OT1)이고 다른 하나는 Cork Encoding이라고 알려져 있는 (주로 유럽어를 위한) TEXtext extended encoding(T1)이다. T1은 OT1의 확장이므로 OT1의 127문자 영역은 공통된다.[3]

글꼴가족(family) 글꼴 제작사에서 같은 부류의 폰트들을 묶어서 부르는 명칭이다. 보통 roman upright, italic, bold 등을 같은 명칭 아래 묶어서 하나의 font family로 한다. Computer Modern Roman은 하나의 폰트 패밀리이다.

계열(series) 폰트의 무게(weight)와 장평(extension)은 series로 분류된다. LATEX에서 흔히 쓰이는 글꼴들은 m(medium), b(boldface), bx(boldface extended), sb(semi-bold), c(condensed) 등의 series를 정의하고 있다. 그러나 모든 폰트가 이런 시리즈를 모두 가진 것은 아니다.

형태(shape) 같은 패밀리에 속하는 폰트들의 모양(upright 또는 roman, italic, slanted 또는 oblique, small caps 등이 폰트의 shape로 정의된다.

크기(size) 폰트 사이즈는 길이단위로 정의된다. 예컨대 10pt, 3mm 등이다. 폰트의 사이즈란 주어진 폰트의 디자인 사이즈를 가리키는 것으로 실제 식자되는 글자의 크기는 이와 다르다.

이 다섯 가지 속성을 모두 지정해야 특정 폰트를 가리킬 수 있다. 예를 들면, `OT1/cmr/m/n/10pt`라는 것은 Computer Modern Roman 글꼴 10포인트를 지칭하는 것으로, `cmr10`이라는 폰트에 할당되어 있다. `cmr10.mf` 또는 `cmr10.pfb`가 실제의 폰트이고 이것을 LATEX에서는 `OT1/cmr/m/n/10pt`로 참조하는 것이다. 같은 폰트라도 사이즈를 달리하여 부를 수 있다. `OT1/cmr/m/n/20.44pt`

2) `http://faq.ktug.or.kr/faq/NFSS`

3) 문자 인코딩은 특정 문자를 이진 부호로 대응시키는 규칙을 가리키는 것으로 폰트 인코딩과는 다른 개념이다. LATEX에서 문자 인코딩은 input encoding이라 하여 inputenc 패키지로 설정하고 폰트 인코딩은 font encoding이라 하여 fontenc 패키지가 마련되어 있다. 이 글에서 다루는 것은 폰트 인코딩이다.

는 같은 폰트를 20.44포인트로 호출한 것이라 생각하면 된다. 또, 기울인 글꼴 호출 OT1/cmr/m/it/10pt는 cmri10.pfb라는 폰트에 할당되어 있기 때문에 itshape에 실제로는 "다른 글꼴"을 불러들여 식자하는 셈이다. 물론 cmr10과 cmri10은 같은 "가족"이기는 하다.

2.2.2 폰트 지정 매크로

NFSS에서 특정 폰트를 선택하게 하기 위한 매크로는 다음과 같은 종류들이 있다.

인코딩 선택 \fontencoding 인코딩을 선택하게 한다.

글꼴가족 선택 \fontfamily 글꼴 가족 명칭을 인자로 취한다. 글꼴 가족 명칭은 미리 알아두어야 한다. \fontfamily{ptm}

계열 선택 \fontseries 시리즈 명칭을 인자로 취한다. 시리즈 명칭은 .fd 폰트 정의 파일을 통하여 \DeclareFontShape라는 별도의 매크로로 미리 LaTeX에게 알려져 있어야 한다. \fontseries{bx}

형태 선택 \fontshape 역시 미리 알려져 있는 폰트 형태 선택자(대개 하나 또는 두 개의 알파벳 문자)를 인자로 취한다. \fontshape{it}

크기 선택 \fontsize 이 매크로는 인자를 두 개 취하는데 앞의 것은 폰트의 자면 사이즈이고 두번째 것은 \baselineskip 값이다. 행간격이 폰트 자체의 속성임에 주의하라. \fontsize{20.44}{24}

일괄 변경 매크로 \usefont encoding, family, series, shape를 한꺼번에 바꿀 수 있다. \usefont{OT1}{cmss}{bx}{it}와 같이 네 개의 인자를 취한다.

폰트 선택 활성화 \selectfont 앞서 정의한 폰트 정의 변경 명령이 실제로 활성화되도록 하는 매크로이다. 예를 들어 폰트의 계열을 바꾸려 한다면 \fontseries{bx}\selectfont와 같이 하여야 효과가 발생한다.

실제로 이와 같은 매크로를 사용자가 문서에서 사용할 일은 거의 없다. 폰트 패키지의 제작자가 이 사항들을 미리 설정해서 제공할 것이기 때문이다.

2.2.3 Default 매크로

사용자가 폰트를 변경하거나 설정하려 할 때 위의 폰트 지정 매크로를 직접 사용하지 않는다. 그 대신 몇 가지 "기본값" 매크로에 할당된 값을 바꾸는 방법을 사용한다.

글꼴 선택 default 가장 중요한 것은 다음 세 개의 \...default들이다. 등호 오른쪽에 이 default 매크로의 default 값을 보였다.

- \encodingdefault = OT1
- \familydefault = \rmdefault
- \seriesdefault = \mddefault
- \shapedefault = \updefault

family 디폴트 한편, font family는 다음 세 가지 디폴트가 정의되어 있다. 역시 등호 오른쪽에 있는 것은 이 default 매크로의 default 값이다.

- \rmdefault = cmr
- \sfdefault = cmss
- \ttdefault = cmtt

위의 \familydefault에 \rmdefault가 할당되어 있었던 것을 보았다.

series 디폴트 \seriesdefault는 다음 두 가지가 정의되어 있다.

- \mddefault = m (medium)
- \bfdefault = bx (boldface extended)

shape 디폴트 fontshape에 관련된 \...default 매크로는 다음과 같다.

- \updefault = n (normal)
- \itdefault = it (italic)
- \sldefault = sl (slanted)
- \scdefault = sc (small capital)

normalfont 이밖에 \normalfont라는 기본값 매크로가 있다. 이것은 문서가 시작하는 시점(\begin{document})에 호출되는 매크로로서, 문서의 글자 기본 선택값을 활성화하고 몇 가지 간격을 설정한다.

2.2.4 사이즈

글꼴 크기는 위의 방식과는 조금 다르게 설계되어 있다. 우선 글꼴 크기 선택 매크로는 다음 아홉 가지(또는 열 가지)이다. memoir 클래스에서 사용하는 열 가지 매크로를 큰 것부터 순서대로 보이면 다음과 같다.[4)]

4) memoir가 표준 LaTeX 클래스와 다른 점 하나는 \HUGE 명령이 더 있다는 것인데, 이 매크로는 memoir뿐 아니라 moresize와 같은 패키지로도 얻을 수는 있다. 다만, memoir의 \HUGE는

표 1: memoir의 폰트 사이즈 매크로 기본값

매크로	기본값	예시
\tiny	6pt	무궁화
\scriptsize	7pt	무궁화
\footnotesize	8pt	무궁화
\small	9pt	무궁화
\normalsize	10pt	무궁화
\large	10.95pt	무궁화
\Large	12pt	무궁화
\LARGE	14.4pt	무궁화
\huge	17.28pt	무궁화
\Huge	20.74pt	무궁화
\HUGE	24.88pt	무궁화

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

이 매크로들에 할당된 "실제 크기"가 얼마인가 하는 것은 전적으로 클래스 디자인에 따른다. 예컨대 표준 LaTeX 클래스에서는 세 개의 폰트 사이즈 설정 옵션([10pt], [11pt], [12pt])이 제공되고 이 각각의 옵션에 따라 위의 매크로에 정의된 "실제 크기" 값이 다르다. 문서 클래스 자체의 디폴트 옵션은 [10pt]인데, 이 조건 하에서 \small은 9pt, \Large는 12pt로 미리 정의되어 있다. memoir에서 폰트 크기 매크로의 실제 값은 표 1을 보라. 이것은 클래스의 글자 크기 옵션 [10pt]인 경우이다. memoir는 표준 LaTeX 클래스보다 더 많은 글자 크기 옵션을 제공한다.

2.3 한글 LaTeX과 NFSS

세 가지 중요한 한글 LaTeX 시스템 —hLaTeXp, HLaTeX, hangul-ucs— 들이 NFSS를 어떻게 수용하였는지 알아보자.

표준 클래스의 \Huge와 크기가 같고 memoir \Huge가 표준 클래스 \LARGE와 \Huge의 중간 크기로 되어 있다.

hLaTeXp 매우 독특한 개념으로 이 문제에 접근하였다. 말하자면, 영문 글꼴 매크로를 "확장"하였다고 할 수 있는 것인데, 예를 들어 cmr이라는 Computer Modern Roman 글꼴이 OT1/cmr/m/n으로 사용되고 있는 것이 문서의 기본값임을 이용해서, 한글 글꼴 명조체를 cmr 글꼴에 붙이는 것이다. 그 이후로, 모든 OT1/cmr/m/n 호출은 한글에 대해서 명조체를 부르는 것과 같이 되도록 하였다. 영문 글꼴과의 대응은 사용자가 임의로 조절할 수 있었는데 일반적으로 그런 귀찮은 작업 없이 기본값을 사용했다는 전설이 있다. NFSS 매크로는 영문의 것을 그냥 사용하고, 그 매크로에 의하여 호출되는 영문자 글꼴에 "대응하는" 한글 글꼴이 자동적으로 설정되도록 하였다고 생각하면 될 것이다. 이 방식의 좋은 점은 역시 한글을 위한 별도의 설정이 없어 번거로움이 없었다는 것이겠고 단점이라면 영문 글꼴이 오직 CM뿐인 경우에만 제대로 동작한다는 문제가 있을 것 같다. 다른 영문 글꼴과 결합하려면 tfm 파일을 또다시 조성해야 하였을 것이기 때문이다.

HLaTeX HFSS (Hangul Font Selection Scheme)이라는 새로운 개념을 도입하였다. 즉, 영문자에는 NFSS를, 그리고 한글과 한자 및 한글 "상징" 문자에는 HFSS를 적용하여 이원적인 폰트 선택 체계를 갖추게 되었다. 이로써 영문 글꼴과 한글 글꼴이 따로 놀게 된다. 예를 들면 \hfontseries, \hfontfamily, \hfontshape와 같은 \h... 매크로가 별도로 제공되었다. 그러나 편의를 위해서, 예컨대 \rmfamily를 부르면 동시에 \mjfamily가 함께 불리도록 하여 영문과 한글을 일일이 별도로 설정해주어야 하는 불편을 덜었다. 아무튼 한글 글꼴 가족 디폴트 매크로로 \mjdefault, \gtdefault, \tzdefault가 HFSS의 근간을 이루고 있었다. 다만, Lambda 체계에서는 단일 글꼴 접근을 취하였기 때문에 별도의 HFSS가 필요하지 않았다.

hangul-ucs 가장 늦게 발달한 hangul-ucs는 앞선 두 매크로 시스템의 장점을 결합하려 하였다. HLaTeX의 방식은 새롭고 영문자와 한글 문자를 분리할 수 있어서 편리한 점이 많기는 하였지만 새로운 매크로의 도입으로 문서의 호환성이 낮아지는 문제가 발생하였다. 그래서 글꼴 선택 스킴은 내부적으로 HLaTeX에서와 같이 영문자 및 한글 글꼴을 분리하되, 사용자 레벨에서는 새로운 매크로 추가를 최소화하여 기존의 NFSS 글꼴 선택 명령으로 한글 글꼴도 선택할 수 있도록 만들었다. 다만 HLaTeX의 중요한 장점이었던 한글 글꼴 가족의 별도 설정 기능은 지원할 수 있게 하였다.

2.4 사용자 수준 폰트 선택 명령

사용자가 문서를 작성할 때 위에서 언급한 세세한 폰트 선택 명령을 쓰게 한다면 번거로움을 이길 수 없을 것이다. 그래서 대부분의 폰트 관련 패키지들은 몇 가지 간단한 사용자 대화 수준의 매크로를 정의해두고 그것만으로 충분히 문서를 작성할 수 있게 하고 있다. 폰트 패키지나 스타일을 설계하는 경우가 아니라면 이 명령만을 익히는 것으로 충분하다.

2.4.1 인코딩 선택

폰트 인코딩은 아무 것이나 선택할 수 없다. 왜냐하면 폰트 자체의 설계와 관련되어 있기 때문이다. 예컨대 Computer Roman 글꼴은 OT1 인코딩으로 되어 있다는 점이 이 폰트 자체의 속성이다.

```
\usepackage[T1]{fontenc}
```

그런데, Computer Modern 글꼴과 같은 경우 너무나 일반적인 글꼴이기 때문에 여기에 대응하는 T1 인코딩의 글꼴이 미리 마련되어 있고, 이것을 EC 글꼴이라고 부른다. 위의 문장은 OT1의 CM 대신 T1의 EC 글꼴을 쓰도록 하는 설정이다. EC 글꼴은 다시 cm-super라고 불리는 폰트 패키지를 이용하도록 되어 있는데 이것은 TEX 배포판의 선택이다.

T1 인코딩의 글꼴 중에서 또다른 유명한 것으로 lmodern이라는 글꼴이 있다. cm-super에 비하여 더 나은 품질인 것으로 알려져 있는데 이 글꼴을 EC 글꼴에 대응하여 쓰려면,

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

이런 식으로 preamble에 적어준다. 이밖에, txfonts와 같이 스스로 T1 인코딩을 설정하도록 설계된 폰트 패키지도 있다.

2.4.2 글꼴 가족 선택

글꼴 가족은 실제 폰트의 명칭을 rm, sf, tt, 세 개의 "기본 가족"에 할당하는 방식으로 먼저 정의한다.

표 2: 글꼴 속성 선택

series	선언형	명령형	shape	선언형	명령형
medium	\mdseries	\textmd{}	upright	\upshape	\textup{}
bold face	\bfseries	\textbf{}	italic	\itshape	\textit{}
			slanted	\slshape	\textsl{}
			small caps	\scshape	\textsc{}

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

위의 설정은 ptm이라는 폰트 이름을 가진[5] times 글꼴을 \rmdefault에 할당하고, phv라는 이름의 helvetica 글꼴, pcr이라는 이름의 courier 글꼴을 각각 \sfdefault와 \ttdefault에 할당한 명령이다. 일단 이렇게 실제 글꼴이 할당된 뒤에는 \rmfamily, \sffamily, \ttfamily라는 \...family 매크로를 쓸 수 있다. 사용자는 사실 이 세 개의 매크로만을 사용하는 것이 옳고, 앞서 소개한 \...default 매크로 정의는 스타일 패키지에서 해주는 것이 일반적이다.

TextFontCommand 그런데 위의 \...family 매크로는 이른바 "선언형"이다. 그래서 "명령형" 매크로도 함께 제공되는데 그것은 각각 \textrm, \textsf, \texttt라는 이름을 가지고 있다.[6]

2.4.3 기타 속성 선택

series와 shape는 표 2에 요약한 바와 같다. 폰트의 크기에 대해서는 이미 7 페이지 §2.2.4 사이즈에서 보인 매크로들을 사용한다.

5) TEX의 폰트 이름짓기는 그 자체가 또하나의 논의 주제이다. 이 글에서는 이미 이름이 지어진 폰트가 주어져 있다고 가정하겠다. 즉 TEX이 사용할 수 있는 times 글꼴의 명칭(TEX fontname)이 ptm이라고 가정하기로 한다. 물론 utm일 수도 있다.

6) 선언형은 인자 없이 주어진 매크로가 실행되는 데 그치지만 명령형은 글꼴 선택 명령의 유효 범위를 인자로 주어진 텍스트로 제한할 수 있다.

2.5 TEX 시스템에 새로운 폰트를 알려주기

> 이 절에서 다루고자 하는 것은 폰트를 "설치"하는 과정이다. 최근의 TEX 배포판들은 폰트 설치 부분에 있어 이전과 비교할 수 없을 정도로 간편하고 쉬워졌다. 이 글에서는 Windows 시스템의 MiKTEX과 Linux/Mac 등의 표준 TEX implementation인 teTEX을 기준으로 설명하려 한다.

TEX이 사용하는 폰트의 종류는 크게 몇 가지가 있다.

METAFONT TEX의 기본 글꼴이다. 그러나 최근 pdf 중심의 작업 과정에서는 그다지 환영받지 못하는 듯하다. METAFONT 자체는 윤곽선 글꼴이지만 pdf에 포함될 때는 pk 폰트라 불리는 pixel bitmap 글꼴로 변환되기 때문이다. pk 폰트는 대부분의 dvi 드라이버들이 화면에 보여주거나 프린트하기 위해서 윤곽선 글꼴로부터 조성하는 비트맵 글꼴인데 해상도에 따라 일일이 만들어내어야 한다.

PostScript (type1) 이것은 Adobe사의 PostScript 기술을 이용한 윤곽선 글꼴이다. .pfb라는 확장명을 가지며, 현재 가장 일반적으로 사용되는 폰트가 되었다.

Truetype Windows 폰트로 알려진 트루타입은 type1 글꼴과 같은 윤곽선 글꼴이지만 구현 방법이 다르다고 한다. 한글 글꼴은 대부분 트루타입이다. TEX쪽에서는 dvi 드라이버 가운데 dvips가 이 폰트를 처리하지 못한다. DVIPDFM*x*는 트루타입을 잘 처리하며 pdfTEX은 트루타입 처리 능력이 있으나 한글 트루타입과 같은 큰 규모의 폰트를 pdfTEX에게 처리하게 하려면 약간의 설정이 필요하다.

Opentype 차세대 글꼴이라 하는 포맷인데, DVIPDFM*x*만이 이 글꼴을 제대로 처리할 수 있다.

TEX에서의 폰트 처리 과정을 그림 1에 나타내었다. 이 글에서는 주로 type1 포스트스크립트 폰트와 트루타입 폰트만을 생각해보고자 한다.

2.5.1 tfm 파일

.tfm란 무엇인가? 이것은 "tex font metric" 글꼴을 가리키는 것으로 TEX이 식자(조판)를 하기 위해서 필요로 하는 최소한의 정보를 담고 있는 글꼴 정보 파일이다. 잘 알려진 대로 TEX은 글꼴 자체를 직접 타입세팅에 이용하지 않으며, 각각의 글자를 모두 하나의 박스로 취급한다. TEX에게 필요한 정보는

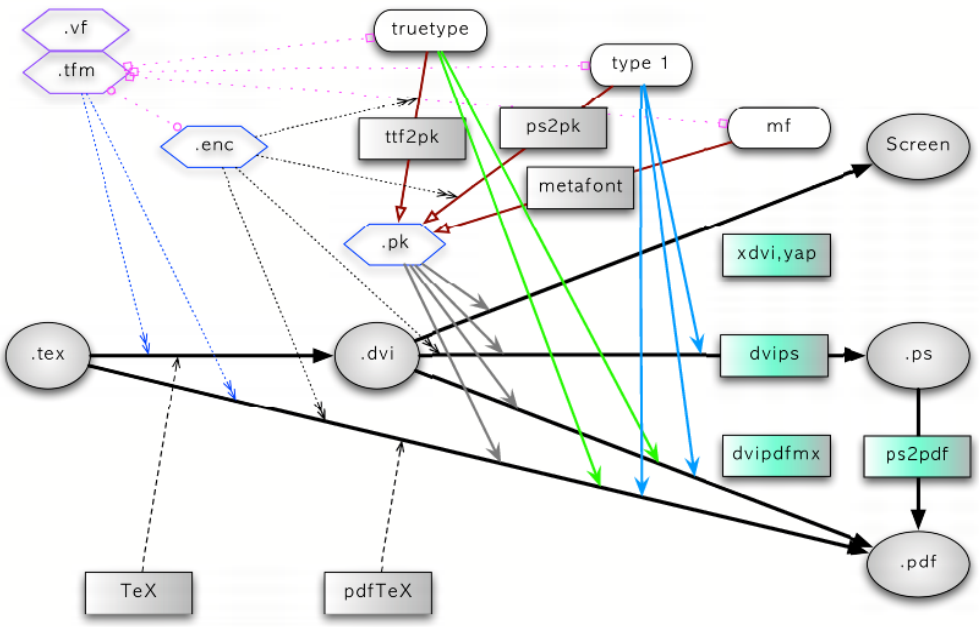

그림 1: TEX의 폰트 처리 과정

이 박스의 폭, 높이, 깊이에 대한 것뿐이다. 높이와 깊이가 별도로 필요한 이유는 baseline에 맞추어서 식자하게 하기 위해서이다. 즉 TEX은 `.pfb`나 `.ttf` 폰트를 필요로하는 것이 아니라 `.tfm`만 있으면 된다. 실제 TEX의 출력물인 `.dvi` 파일에도 폰트의 이름만이 참조되도록 되어 있고 글꼴 그 자체는 포함하고 있지 않다. 글꼴이 들어 있지 않은 `.dvi`에 글꼴 정보를 넣어서 사람이 알아볼 수 있는 형태, 즉 스크린 디스플레이나 인쇄 지면 상의 점으로 변환하는 일은 dvi 드라이버가 하는 일이다. 그러므로 어떤 폰트든지 그것을 TEX에서 사용하려면 `.tfm` 폰트를 만들어서 TEX의 처리 과정에 제공해주어야 한다. `.tfm`을 제작하는 것은 폰트 디자이너나 패키지 작성자가 할 일이므로 우리는 단순히 이 폰트 메트릭이 주어져 있다고 가정하겠다. 사실 한글 글꼴의 경우는 일이 그렇게 한가하지 않아서 사용자 자신이 `.tfm`을 직접 만들어야 하는 경우도 많지만 이 글에서는 논외로 하자.

2.5.2 enc 파일

원래 .enc 파일은 글꼴의 인코딩 map이라 할 수 있는 것이다. dvips가 포스트스크립트 폰트를 이용할 때 쓰는 경우가 있고, 일반적인 인코딩이 아닌 특별한 인코딩을 가진 폰트의 경우 매우 중요하게 취급될 수도 있다. 한글 글꼴에 있어서 이 enc 파일은 주로 pdfTEX에게 한글 서브 폰트 글꼴에 대한 정보를 알려주기 위하여 사용된다. 유니코드 폰트를 이용하는 한글 글꼴의 경우 기술적으로 이 파일이 크게 필요하지 않고 실제로 DVIPDFM*x*는 별도의 enc 파일 없이 트루타입 등을 처리하지만 현재 버전(1.2x)의 pdfTEX은 아직까지 한글 서브폰트 트루타입을 처리하려면 enc 파일이 요구된다. 이 파일 역시 사전에 준비해두어야 하는 것으로 사용자가 신경쓸 부분이 아니다. 역시 이 파일들은 이미 준비되어 있다고 가정하겠다. 다행히 다음 버전의 pdfTEX은 한글 글꼴과 같은 CID 폰트에 대해서 이러한 enc 파일을 별도로 만들지 않아도 된다고 하는데, 역시 사용자가 크게 문제삼을 만한 일은 아닐 것이다.

2.5.3 폰트 파일 자체

폰트 파일 자체가 없다면 아무리 tfm 등의 파일이 있어도 당연히 그 글꼴을 쓸 수 없다. 트루타입 글꼴은 시스템의 폰트 디렉터리에 있는 경우도 많다. 폰트 파일 자체에 대한 정보를 미리 알아두는 것은 나중을 위해서도 나쁘지 않다.

2.5.4 subfont란 무엇인가?

유니코드 UCS 인코딩의 경우 현대 한글 음절 문자 영역은 [AC00]에서 [D7AF]까지 모두 11,172자이다. 한글 관련 영역이 더 있지만 우선 이것만 생각해보면, 음절 단위로 한글만 하나의 글꼴로 만들면 무려 1만여 자면을 갖는 폰트가 만들어진다. TEX은 설계상 하나의 폰트에 127글자가 있는 것으로 보고 타입세팅을 하고, T1 인코딩까지 확장한다해도 256글자까지밖에는 한 바이트로 다룰 수 없다. 즉 폰트캐릭터 레지스터가 1바이트인 이상 그 글꼴의 256번 밖의 문자는 TEX에서 접근할 수가 없는 것이다. 이 한계를 극복하기 위하여 Omega의 경우 폰트캐릭터 레지스터를 2바이트로 확장하여 UCS 글자를 다루려 시도하였는데, 우리는 여전히 LATEX으로 이 많은 문자를 가진 폰트를 어떻게 처리할 것인가 생각해야 한다.

그래서 나온 게 subfont라는 아이디어였다. 이것은 하나의 글꼴을 256자씩 여러 개의 부속 폰트 파일로 쪼개어서 취급하자는 것이다. 이를테면 TEX에게

는 마치 서로 다른 폰트인 듯이 각각의 서브폰트를 알려주되 나중에 subfont들을 하나의 단위로 취급하도록 해주면 되지 않겠는가? 다행히 요즘 TEX은 등록할 수 있는 폰트의 개수가 256개로 제한되지 않기 때문에 수많은 서브폰트라 해도 TEX이 그것을 처리할 능력을 갖추게 되었다고 할 수 있다. 그 결과, 한글 영역의 `.tfm` 글꼴들은 예컨대 `XXXac`에서 `XXXd7`까지 수십 개의 서브폰트 형식으로 제공되게 되었다.

2.5.5 map 파일

dvi 드라이버들은 `.dvi` 파일을 목적 포맷(스크린, 특정 프린터, pdf 파일 등)으로 변환하기 위해서 폰트 정보를 필요로 한다. 대표적인 세 가지 폰트 map 파일은 각각 dvips, pdftex, dvipdfm을 위한 것들이고 다른 드라이버들은 이 포맷의 map 파일을 가져다가 이용하는 경우가 많다. 혹은 yap과 같이 자신은 아예 폰트에 대해서 전혀 상관하지 않으면서 없는 글꼴이 나오면 `mktexpk`라는 프로그램을 불러서 pk 폰트 파일을 만들어달라고 요청하는 프로그램도 꽤 된다. `mktexpk`는 `mf`, `gsftopk`(`ps2pk`), `ttf2pk` 등을 차례로 실행해보면서 성공하는 글꼴이 있는지를 찾아서 pk 비트맵 폰트를 만든다. 그러면 yap은 만들어진 pk 글꼴을 가져다가 화면에 뿌려준다. DVIPDFM*x*는 dvipdfm의 map 파일을 그대로 이용한다.

updmap 유틸리티 폰트 제작자가 이 모든 경우의 map 파일을 모두 만들어 제공하기는 어렵다. 그러므로 예컨대 dvips용 map 파일만 주어져 있는 경우에 다른 드라이버가 사용할 수 있도록 이 map 파일로부터 각 드라이버별 map 파일을 만들어주면 편리할 것이다. 예전에는 이것이 자동화되어 있지 않아서 드라이버별로 일일이 수작업으로 map 파일을 편집해야 하는 때도 있었다. 다행히 이 일을 자동화해주는 스크립트가 있는데 그것이 updmap이라는 것이다. 이 유틸리티는 dvips용 각 폰트별 map 파일을 읽어서 dvips, pdftex, dvipdfm용 map 파일을 일괄로 만들어낸다. updmap이 만들어내는 map 파일은 teTEX의 경우 각각 `psfonts.map`, `pdftex.map`, `dvipdfm.map`이라는 이름을 가지게 되어 있고, 모든 드라이버들은 이 가운데 하나를 자신의 설정 파일에 "필수적으로" 불러들이도록 설정한다. MiKTEX의 경우는 유별나서 이 모두가 전부 `psfonts.map`이라는 이름을 갖는데, 위치한 디렉터리가 어디인지에 따라 구별하여 불러들인다. 무슨 파일을 불러들이든 그것은 TEX 배포판에서 설정하기 나름이고 사용자는 거기에 신경쓰지 않아도 된다.

updmap이 어떤 map 파일들을 읽어서 psfonts.map 등을 만들 것이냐는 updmap.cfg라는 설정 파일에 달려 있다. 이 파일은 시스템 내에서 여러 개 있을 수도 있는데, 그 중의 하나만이 "유효"하다. 사용자는 이 "유효"한 파일의 위치 정도를 확인해두는 것으로 충분할 것이다. home 디렉터리에서 다음 명령을 실행하면 그 위치를 알 수 있다.

```
$ kpsewhich --format="web2c files" updmap.cfg
```

updmap을 실행함에 따라 유효한 파일의 위치가 달라지기도 하므로, 만약 이 설정 파일에 어떤 작용을 가하려면 이 위치를 그때그때 확인하는 것이 좋다. type 1 포스트스크립트 폰트를 시스템에 설치할 때 이 유틸리티를 이용하여 폰트 설정을 하는 것이 표준이다.

2.5.6 fd 폰트 정의 파일

앞서 4 페이지 §2.2에서 본 바대로, TEX에게는 사용할 폰트에 대한 정보가 미리 주어져 있어야 한다. *encoding, family, series, shape* 등에 대한 정보를 제공하려면 .fd 파일을 이용한다. 이 파일에는 최소한 font family 선언과 series, shape 선언이 포함되어 있어야 하고 파일 이름이 일정한 규칙을 따르게 되어 있다. 즉, <ENC><FAM>.fd라는 파일 이름을 가져야 하는 것이다. TEX이 OT1/cmr/m/n/10pt라는 글꼴을 요청받으면 OT1cmr.fd라는 파일을 찾도록 되어 있는 것이다. 이러한 설정은 plainTEX의 것이 아니고 LATEX 특유의 방식이다. 이 파일의 내용은 매크로의 일종으로 취급된다. 또한, fd 파일과 함께 사용자가 편리하게 \usepackage 문장만으로 해당 폰트를 사용할 수 있도록 지원하기 위한 스타일(.sty) 파일을 함께 제공하는 경우도 많다.

2.5.7 기타

그밖에 폰트 패키지에서 제공되는 파일로 virtual fonts (vf) 파일,[7] type 1 포스트스크립트 폰트의 폰트 메트릭인 afm 파일 등이 있다.

7) vf 파일은 기존 폰트 파일로부터 새로운 가상 폰트를 만들어내는 데 사용된다. 예컨대 자소만 포함하고 있는 UHC 글꼴의 .pfb로부터 한글 인코딩에 상당하는 wmj 폰트를 만들어내기 위해서 가상 글꼴이 쓰인 것이 한 예다.

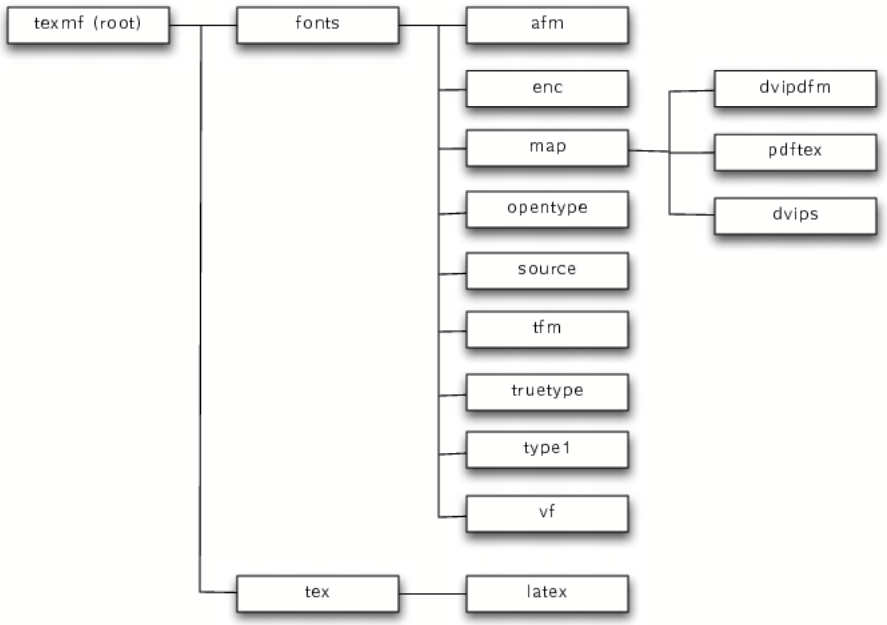

그림 2: teTEX의 폰트 관련 TDS

2.6 폰트 파일의 자리 찾기

> TEX 시스템에서 해당 파일들이 놓여야 할 위치를 알아둔다. 이른바 "texmf tree 구조"라고 하는 이 위치는 TEX 시스템을 운용하는 데 매우 중요하다.

TEX은 엄청나게 방대한 시스템이라 이 시스템을 효율적으로 관리하기 위해 파일들이 놓여야 할 상대 위치를 표준화해두고 있다. 이것을 TDS (TeX Directory Structure)라고 한다. MiKTEX 현재 버전(2.4)은 teTEX과는 다른 디렉터리 구조를 채택하고 있으므로 주의하여야 한다. 그림 2와 그림 3은 폰트와 관련된 TDS의 일부를 보여주는 그림이다.[8)]

MiKTEX의 TDS에서 주의할 것은, map 파일을 위한 별도의 디렉터리가 없다는 점이다. 그러므로 map 파일들은 `texmf/{dvipdfm,pdftex,dvips}` 아래

8) 이 글에서 teTEX이란 teTEX 3.x 버전, kpathsea 라이브러리 4.x 버전을 기준으로 하고 있다. MiKTEX은 2.4

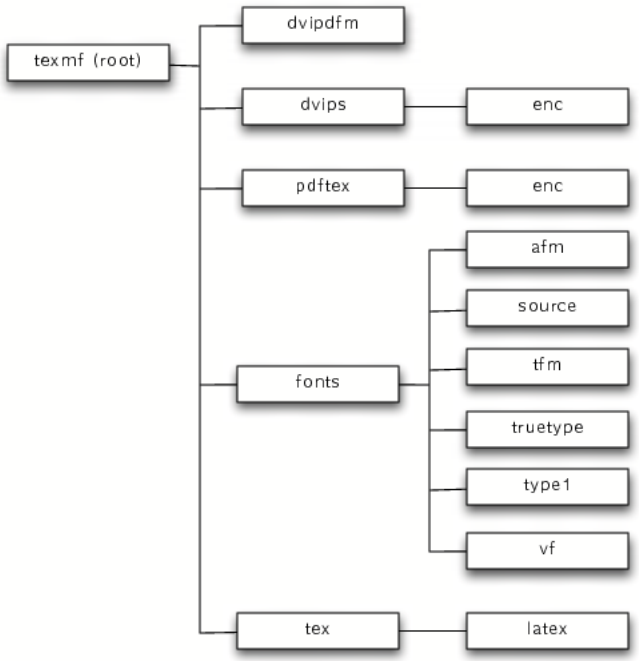

그림 3: MiKTEX의 폰트 관련 TDS

의 적당한 곳에 두면 된다. `texmf/fonts` 아래에 map 파일이 있으면 찾지 못한다. 또한 MiKTEX은 opentype을 위한 디렉터리도 없다. 이 말은 즉, MiKTEX 시스템에서는 opentype 폰트를 쓰지 못한다는 뜻이다. 다음 버전에서는 이러한 시대에 뒤진 특징이 개선되리라고 기대한다.

MiKTEX 사용자를 위해서 teTEX 기준의 폰트 패키지의 파일들을 어떻게 옮겨야 할지 생각해보자. 다음과 같은 규칙을 따르면 될 것이다. ttf2pk에 관한 것은 그림에는 나와 있지 않지만 한글 폰트 패키지에서는 흔히 볼 수 있으므로 여기서는 그것도 고려하기로 한다.

```
texmf/fonts/map/dvipdfm// -> texmf/dvipdfm//
texmf/fonts/map/pdftex//  -> texmf/pdftex//
texmf/fonts/map/dvips//   -> texmf/dvips//
texmf/fonts/map/ttf2pk//  -> texmf/ttf2pk//
texmf/fonts/enc/pdftex//  -> texmf/pdftex/enc//
texmf/fonts/enc/dvips//   -> texmf/dvips/enc//
texmf/fonts/enc/ttf2pk//  -> texmf/ttf2pk/enc//
```

위의 표에서 `texmf`라고 한 것은 실제로는 `texmf-local`일 수도 있고 MiKTEX

의 경우 localtexmf일 수도 있다. 그리고 //로 표시한 것은 그 아래의 어느 위치든 찾을 수 있다는 뜻이다. 되도록이면 관리의 용이함을 위해서 아래에 폰트 제작사나 이름에 상당하는 디렉터리를 만들고 거기에 넣어둔다.

2.7 kpathsea에게 파일 목록 알려주기

TEX 시스템은 kpathsea라는 라이브러리를 이용해서 파일을 찾는다. 파일 위치를 찾아서 되돌려주는 kpathsea가 적절하게 작동하려면 미리 파일 목록 데이터베이스를 만들어두어야 하는데, 새로운 파일이 추가되면 이 목록 데이터베이스를 갱신해주어야 할 것이다. 이것은 mktexlsr 또는 texhash라는 프로그램을 실행하면 된다. MiKTEX에서는 initexmf 유틸리티에 -u (update-fndb) 라는 옵션을 주어서 실행하는 것이 여기에 해당한다. texmf 트리에 새로운 파일이 추가되면 이 절차가 항상 필요한데, 윈도용 MiKTEX은 정말 윈도답게 MiKTeX Options라는 GUI 프로그램의 <Refresh> 단추로 이 기능을 실행하게 하고 있기도 하다.

```
$ mktexlsr
```

2.8 글꼴 설정하기

이제 각 dvi 드라이버들을 위한 설정을 해주어야 한다.

2.8.1 type 1 글꼴

type 1 포스트스크립트 글꼴은 보통 dvips용 map 파일과 함께 제공된다. 이 map 파일을 updmap을 이용하여 등록하면 된다. teTEX의 경우 다음 명령을 실행하는 것으로 충분하다.

```
$ updmap --enable Map=XXX.map
```

시스템에 따라 updmap 대신 시스템 관리자 권한으로 updmap-sys를 실행하는 경우도 있으며, Map 지시자 대신 MixedMap을 쓸 수도 있다. Map과 MixedMap의 차이에 대해서 이 글에서는 별도로 언급하지 않는다. 다만 대개 Map 지시자를 쓰면 큰 문제 없다고만 말해두기로 하자.

윈도의 MiKTEX은 이와 조금 다른 방법을 쓰는데, updmap.cfg를 추가하는 것이다. 예컨대 TEX 시스템을 보통 권장하는 방법대로 C:\TeX\ 아래 오도록, C:\TeX\localtexmf와 \C:\TeX\texmf로 설치하였다고 하자. MiKTEX의 main tree는 사용자가 건드려서는 안된다. 왜냐하면 MiKTEX 자신이 온라인 업그레이드를 하면서 지속적으로 덮어쓰기 때문이다. 모든 사용자 설정은 localtexmf에서 하여야 한다. 이제 C:\TeX\localtexmf\miktex\config 폴더를 찾아 열어서 여기에 updmap.cfg라는 파일을 하나 만들자. 내용은 다음과 같이 한다.

```
$ ‘kpsewhich --format="web2c files" updmap.cfg‘

Map XXX.map
```

첫번째 행은 중요하다. 이 행을 빠뜨리면 다른 폰트를 전혀 사용할 수 없는 상황이 생길 수도 있다.

이 후에 다음 명령을 명령행에서 실행한다.

```
#> initexmf -u
#> initexmf --mkmaps
```

두 번째 명령은 옵션 인자에 --가 두 개 붙어 있음에 주의하라. 파일의 위치가 잘못되거나 편집 중에 오타가 발생하면 아무런 에러 메시지도 보여주지 않지만 폰트를 사용할 수 없을 따름이다.

2.8.2 truetype 한글 글꼴

한글 글꼴은 대부분 트루타입으로, 윈도 글꼴이다. 현재 GNU GPL 라이센스로 배포되고 있는 한글 트루타입은 은글꼴이 거의 유일하며[9], 공개 글꼴이라 할 수 있는 한겨레결체를 포함하더라도 몇 종류 되지 않는다.

한글 트루타입을 hangul-ucs에서 사용하도록 폰트를 조성하고 설정하는 과정에 대해서는 ttf2hlatexfont를 이용하는 방법이 잘 알려져 있으므로 이 글에서 다루지는 않는다. 이미 폰트가 조성되어 있다면 이것을 설치하고 설정하는 방법만을 생각해보자.

9) alee 글꼴도 있다.

우선 앞의 단계에서 폰트의 설치는 이미 끝났을 것이다. 트루타입 글꼴을 설정하기 위해서는 updmap을 사용하지 못하는데 그 이유는 dvips가 트루타입을 지원하지 않기 때문이다. dvips는 어차피 pk 폰트로 글꼴을 변환해서 사용할 것이므로 별도로 설정을 해주어야 할 필요가 없다.[10)]

pdfTEX과 DVIPDFM*x*만을 위한 설정을 할 수밖에 없으므로 다음과 같은 과정을 거치면 된다.

dvipdfmx.cfg 설정 파일 DVIPDFM*x*[11)]는 dvipdfmx.cfg라는 설정 파일을 읽어서 dvi 파일을 pdf로 변환한다. 시스템에 둘 이상의 같은 이름의 파일이 있더라도 "유효"한 것은 하나뿐이므로 이 파일의 위치를 다음과 같이 하여 확인해두는 것이 필요하다.

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

옵션 중에 dvipdfm을 dvipdfmx로 잘못 쓰지 않도록 주의한다. 이 파일을 찾아서 마지막에 다음 한 행을 추가하면 된다.

```
f XXX.map
```

map 파일 이름은 대개 cid-XXX.map과 같은 형태를 띠고 있기 쉽다.

ttf2pk.cfg 설정 파일 ttf2pk는 트루타입 글꼴을 pk 폰트로 변환하는 유틸리티이다. dvips나 yap, xdvi 등의 드라이버들은 화면에 pk 글꼴을 뿌려주는 방식으로 dvi 파일을 처리하므로 이 유틸리티가 있어야 화면에서 xdvi로 문서를 볼 수 있다. 별도의 설치를 요하는 것으로 teTEX의 일부가 아니지만 hangul-ucs 설치 패키지는 대부분 이 유틸리티를 함께 설치해줄 것이다. MiKTEX의 경우에는 KTUG patch를 적용해야 한다.

dvipdfmx.cfg의 경우와 마찬가지로 이 파일의 위치를 찾는다.

10) dvips가 트루타입을 지원하지 않는 관계로, latex → dvips → ps2pdf를 거쳐 pdf를 만들지 않으면 아니되는 pstricks나 powerdot 등에는 트루타입을 "원칙적으로" 사용할 수 없다. hangul-ucs의 기본 글꼴이 트루타입인 까닭에 이 문제를 피해가기 위해서 UHC type1을 포팅한 uhc-unicode라는 폰트 패키지를 별도로 배포한다.

11) dvipdfm이 아님에 주의하라. dvipdfm의 트루타입 처리 능력은 상당히 취약해서 DVIPDFM*x*와 비교할 수 없다.

```
$ kpsewhich --format="other text files" --progname="ttf2pk" \
ttf2pk.cfg
```

이 파일을 불러서 마지막에 해당 map 파일을 적어넣는다.

```
map +XXX.map
```

이 map 파일은 대개 `XXX-ttf2pk.map`과 같은 이름을 가지고 있기 쉽다.

pdfTEX의 경우 트루타입을 pdfTEX이 사용할 수는 있지만 별도의 map 파일 설정은 하지 않는다. 어차피 pdfTEX은 별도의 설정 파일이 없기 때문에 굳이 하려면 `pdftex.map` 파일에 설치하려는 폰트의 pdfTEX용 map을 "붙여넣기" 하여야 하는데 이것은 `updmap`이 실행될 때마다 바뀌기 때문에 적절하지 못하다. 차라리 문서에서 명령으로 특정 map 파일의 사용을 지시하는 것이 더 합리적이라고 하겠다.

* * *

이제 설치와 설정이 끝났다. 일반적으로 새로운 글꼴 세트를 설치할 때는 위의 절차를 따르면 된다. 한글 글꼴은 파일 수가 무척 많고 부피가 커서 다루기가 까다롭지만 TEX 시스템을 일관성있게 이해하고 있으면 그다지 어려운 일은 아닐 거라고 생각한다.

3 실전: 문화부 type 1 글꼴 설치하기

문화부 글꼴 또는 문체부 글꼴이라 함은 문화관광부에서 제작하여 배포하였던 한글 글꼴 세트를 가리키는 말이다. 이 글꼴은 원래 원도 자체는 조합형 글꼴로서 자소만을 디자인하고 조합 규칙을 제시한 것에 그쳤지만 그 뒤 역시 문화관광부의 용역으로 서울시스템에서 트루타입으로 제작하여 일반에 공개되었다. 서울시스템은 용역으로 제작·배포한 서체임에도 불구하고 저작권을 자사에 귀속시키는 메시지를 글꼴에 포함함으로써—비록 디자인과 트루타입 제작을 직접 하였다고는 하나—사용자를 우롱한 바도 있으며, EUC-KR 완성형 범위의 자소만을 포함하고 심지어 영문과 한자 영역까지 넣지 않음으로써 사실상 "쓸모없는" 글꼴로 만들어버리는 데도 크게 기여하였다. 상업용으로 판매할 것이 아니라 정부에 납품할 것이라 그렇게 성의없이 만들었는지는 모르겠으나, 아무튼 매우 실망스러운 글꼴이었던 것만은 틀림없으며,

글꼴 디자인의 아름다움을 고려하면 아까운 것이기도 하였다. 현재 이 글꼴은 Adobe사의 ftp에서 cid 형식, opentype 형식으로 다운로드받을 수 있는데, 원래의 트루타입과는 달리 (완성 글꼴에서 온 듯한) 한자 영역을 몇 개의 글꼴이 추가로 포함하고 있다.

이 글꼴의 라이센스는 매우 모호하며, 문화부 스스로가 확정적인 입장을 표명하지 않고 있다. 주무담당자의 견해는 "상업적 용도로 사용되지 않으면" 자유롭게 사용할 수 있다는 것이라고 하나, 이 "비상업적 사용"이라는 조건이 항상 문제가 되었다. 그 "사용"에 글꼴의 임베드, 변형까지 포함하는 것인지 어떤지도 확실치 않다. 그러나 이 정도 품위의 국가가 제작하여 배포한 글꼴에 대하여 까다로운 라이센스를 운위한다는 것은 그 자체로 넌센스일 것이다. 당연히 TEX에서의 사용을 위한 변형은 공정 사용(fair use)의 범주에 든다고 보아야 하지 않을까 한다.

한글 글꼴이 아쉽기만 한 TEX 사용자를 위해서 비록 불만스러우나마 문화부 글꼴의 type1 조성을 시도하였다. 이것을 문화부 글꼴 type1이라 부르기로 한다. 이 글꼴이 가진 특징은 다음과 같다.

(1) type1 PostScript 글꼴이며, subfont 형태로 제공된다. 즉 각각의 `.pfb`들은 256글자씩만을 포함하고 있다.

(2) EUC-KR 완성형 한글 2350자와 한자 4888자만을 포함하고 있다.

(3) HLATEX용은 없으며 오직 hangul-ucs만을 위한 것이다.

이제 이 글꼴을 Windows MiKTEX 시스템에 설치하는 것을 생각해본다. 아무래도 윈도 사용자가 가장 많을 것이며, 또 글꼴 설치 설정에 어려움을 겪을 가능성도 윈도에서 가장 크기 때문이다.

3.1 설치와 설정

3.1.1 다운로드

이 글꼴은 다음 위치에서 다운로드받을 수 있다. `http://faq.ktug.or.kr/faq/Karnes/2006-03` 이 페이지 중간쯤 3월 26일 항목에 "문체부 글꼴 Type1"이 있다. 파일은 두 개인데 취향에 따라 설치하면 되지만 여기서는 둘 다 설치하는 것으로 하자.

MiKTEX은 `C:\TeX` 아래 설치되어 있는 것으로 가정하겠다. 다른 위치에 설치되어 있어도 상관은 없지만 권장하는 위치가 아닌 다른 위치에 설치할 때는 그로 인해 발생하는 모든 문제에 스스로 책임질 자신이 있었기 때문일 것이므로 이 가이드에서 별도로 언급하지는 않는다.

3.1.2 설치

1. 파일을 푼다. 그러면 현재 위치의 아래에 `texmf.local`이라는 폴더가 생겨난다.
2. 파일 또는 폴더를 다음과 같이 이동한다. 이 절차는 MiKTEX 사용자에게만 필요하다. 또한 type1 글꼴이므로 dvips 폴더만 이동하면 된다.
   `texmf.local/fonts/map/dvips -> texmf.local/dvips`
3. texmf.local 폴더 전체를 `C:\TeX\texmf-mhtype1`이라는 이름으로 옮긴다.
4. MiKTeX Options를 실행하여 ROOTS 탭에서 이 폴더를 등록한다.

3.1.3 설정

1. 만약 `C:\TeX\localtexmf\miktex\config\updmap.cfg`가 있으면 이 파일을 편집하고 없다면 `..\localtexmf\miktex\config\updmap.cfg`를 만들어서 필요한 설정을 해둔다. §2.8.1을 참고하여 `$`로 kpsewhich를 부르는 한 행을 추가하면 된다.
2. `..\localtexmf\miktex\config\updmap.cfg`에 다음 행을 추가한다.
   ```
   Map mhtypeone.map
   Map mhtypeone-extra.map
   ```
3. 다음 명령을 차례로 실행한다.
   ```
   #> initexmf -u
   #> initexmf --mkmaps
   ```

3.2 테스트

다음과 같이 하여 설치와 설정이 잘 되었는지를 검토해본다.

- `..\localtexmf\dvipdfm\updmap\psfonts.map` 파일의 내용 중에 `omhmjm` 관련 항목이 있는지를 찾아본다.
- `kpsewhich fd LUCmhmj.fd` 명령을 실행하여 fd 파일이 잘 찾아지는지 확인한다.

4 문화부 type1 글꼴을 이용하여 문서 작성하기

여기서부터는 비단 MiKTEX에 국한되지 않는 일반적인 폰트 활용 방법을 알아본다.

4.1 hangul-ucs의 폰트 선택 체계

hangul-ucs 사용설명서에 폰트 선택 체계에 대해서는 상세히 나와 있다. 간단히 요약하면 다음과 같다.

`\SetHangulFonts` preamble에 쓰여서 문서 전체의 글꼴을 지정한다. 세 개의 인자를 가지며 각각 rm, sf, tt 글꼴에 상당하는 한글 글꼴 family를 지정한다.

`\SetHanjaFonts` preamble에 쓰여서 문서 전체의 글꼴을 지정한다. 세 개의 인자를 가지며 각각 rm, sf, tt 글꼴에 상당하는 한자 및 기호 문자의 글꼴 family를 지정한다.

`\SetSerifFonts` rm 가족에 해당하는 한글 및 한자, 기호 문자 글꼴 family를 선택할 수 있다. 두 개의 인자를 가지며 각각 한글, 한자 및 기호 문자에 해당하는 글꼴 family를 적어준다.

`\SetSansFonts` sf 가족에 해당하는 한글 및 한자, 기호 문자 글꼴 family를 선택할 수 있다. 인자는 `\SetSerifFonts`와 같다.

`\SetMonoFonts` tt 가족에 해당하는 한글 및 한자, 기호 문자 글꼴 family를 선택할 수 있다. 인자는 `\SetSerifFonts`와 같다.

`\SetAdhocFonts` 특정한 주어진 영역의 글꼴 family를 임시적으로 바꾸는 데 쓰이는 명령이다. 이 명령의 유효 범위는 현재의 scope 범위이므로 `\begin`에서 `\end` 범위 안에서 유효하다.

hangul-ucs가 사용하는 글꼴 인코딩은 LUC 인코딩인데 이것은 스타일 자신이 스스로 설정하기 때문에 사용자는 신경쓸 필요가 없다. 그리고 폰트 series와 shape는 NFSS의 매크로를 사용한다. 예를 들어 굳이 폰트 시리즈를 특별히 선택할 경우라면(그럴 가능성은 거의 없겠지만) `\fontseries{bx}\selectfont`와 같이 하면 된다.

`\rmfamily`, `\sffamily`, `\ttfamily` 그리고 `\textrm`, `\textsf`, `\texttt` 들의 폰트 선택 매크로들은 모두 한글 글꼴과 연동된다. LaTeX에서와 같은 `\hfont...` 명령이나 `\textgt` 등은 제공하지 않는다.

4.2 문화부 글꼴 type1의 이해

글꼴을 새로 설치한 후 이것을 사용하려면 해당 글꼴의 특성에 대해서 약간의 이해가 필요하다. 표 3은 문화부 type1 글꼴의 특성을 요약한 것이다.

이 특징 이외에도, 이 글꼴은 c-series 없이 m-series로만 제공된다.

표 3: 문화부 글꼴 type 1의 특징

서체 이름	tfm/pfb 이름	hangul-ucs	bold	slanted	한자
문화바탕체	omhmj{m,b}	mhmj	있음	없음	있음
문화돋움체	omhgt{m,b}	mhgt	있음	없음	있음
문화궁서체	omhgsm	mhgs	없음	없음	없음
문화궁흘림	omhghm	mhgh	없음	없음	없음
문화쓰기체	omhpnm	mhpn	없음	없음	없음
문화쓰기흘림	omhphm	mhph	없음	없음	없음
문화훈민정음	omhmgm	mhmg	없음	없음	없음

4.3 pdfLaTeX 이용하기

pdfLaTeX을 꼭 이용해야 하는 상황은 몇 가지가 있다. 예컨대 beamer, attachfile, leaflet 등 pdfLaTeX에서만 동작하는 패키지가 있기 때문이다. 트루타입도 잘 사용할 수 있지만 매번 map 파일을 적어주어야 하는 것이 문제라면 문제다. 여기서는 문화부 글꼴 type1을 본문 글꼴로 사용해보자.

```
\documentclass{article}
\usepackage{dhucs}
\usepackage{dhucs-ucshyper}
%\usepackage{ifpdf}
\ifpdf
 \usepackage{dhucs-cmap}
 \pdfmapfile{+unttf-pdftex-dhucs.map}
\fi
\SetHangulFonts{mhmj}{mhgt}{untz}
```

4.4 dvips 이용하기

type1 글꼴이 특히 필요한 곳은 dvips → ps2pdf로 작업해야 하는 경우이다. pstricks가 가장 대표적인 패키지이고 psfrag이나 powerdot, 그밖에 이런 루트를 요구하는 클래스와 패키지가 제법 된다. 주로 dvips의 PostScript 처리 능력을 이용하려는 경우이다.

pdf 제작 관련 이슈 가운데 한글 bookmarks는 거의 해결이 되었고 한글 텍스트의 검색과 추출에 관한 것이 남았는데, pdfTeX에 대해서는 dhucs-cmap 패키지를 통해 결말이 이루어졌다. 그러나 dvips 관련 문제는 아직 충분하지

않은 면이 있다. 문화부 글꼴 type1을 사용한 경우 다음과 같은 사실이 알려져 있다.

1. dvips 결과를 Adobe Distiller로 pdf 변환한 파일은 한글 텍스트의 검색과 추출이 가능하였다.
2. dvips 결과를 Ghostscript ps2pdf로 변환한 pdf 파일은 Adobe Reader에서는 텍스트의 검색과 추출이 되지 않았으나 맥의 pdf viewer에서는 가능했다.
3. dvips 결과를 맥의 pstopdf (Apple Distiller)로 변환한 pdf 파일은 텍스트의 검색과 추출은 가능하였으나 하이퍼링크와 책갈피가 유실되었다.

이렇게 써두면 기능이 만족스럽지 못한 듯이 보일는지 모르나, 아예 dvips 이용 경로에서는 텍스트의 검색 추출은 고사하고 한글 책갈피조차도 제대로 되지 않았던 점을 고려해서 위의 결과를 보아주기 바란다. 앞으로 GhostScript의 버전이 높아지면 어쩌면 한글 텍스트 문제는 좀더 쉽게 해결될 수 있을는지도 모른다.

```
\documentclass{article}
\usepackage{dhucs}
\usepackage[dvips]{dhucs-ucshyper}
\SetHangulFonts{mhmj}{mhgt}{mhgt}
\SetHanjaFonts{mhmj}{mhgt}{mhgt}
```

5 결어

이 문서가 도움이 되기를 바란다.