HLaTeX 이후 한글 LaTeX의 발전

Karnes*

2006년 9월

요 약

이 글은 〈TEX과 그 언저리〉라는 글과 함께 2006년 9월의 KLDP F/OSS 컨퍼런스 KTUG BoF 토론 자료로 작성한 것이다. 앞선 글에서 다룬 것이 TEX 시스템의 발전에 관한 것이었다면 이 글은 한글 환경에 관련된 문제를 취급한다. TEX에서 한글을 구현하기 위한 노력의 경과와 더불어, KTUG의 기여에 대해서도 부분적으로 언급한다.

〈 목 차 〉

1 緖

이 글은 특히 2001년 KTUG 사이트가 활동을 시작한 이래 한글 LaTeX 환경이 얼마나 변화했는지를 간략하게 제시하려는 것이 목적이다. 논의의 편의상 모든 저자와 인명에 경칭을 붙이지 않았고, 익숙한 약어는 약어 그대로 처리하였다. 이 글에서 문제삼는 것은

*KTUG member. e-mail: info@mail.ktug.or.kr

"한글의 구현"에 대한 것이다. 그러므로 KTUG이 하고 있는 일 중에서 예컨대 사용자를 위한 설치 지원 등의 문제는 이 글에서 다루지 않는다.

이 글은 운영체제와는 상관없는 주제를 다루고 있기는 하나, 우리가 다룰 주제와 관련해서 주로 문제가 일어나는 곳이 Windows 시스템이기 때문에 부득이 Windows 운영체제에서의 상황만을 언급한 곳이 두어 곳 있다. 독자의 양해를 부탁한다. 이와 관련하여, 이 글을 쓰는 또하나의 목적이 MiKTEX+WinEdt이라는 환경의 한계가 무엇인가를 분명히 하기 위해서임을 밝혀둔다. 이러한 설정으로는 2001년 이전 HLATEX 사용 상태를 2006년인 지금도 여전히 벗어나지 못한다. 무엇을 할 수 있고 무엇을 할 수 없는가, 그 당시와 비교해서 어떤 것이 달라졌는가를, 적어도 알고 있어야만 더 합리적인 선택이 가능할 것이다.

2 HLATEX에서 KTUG까지

2.1 배경

한글을 LATEX에서 사용하려는 시도는 1990년대 초부터 KAIST를 중심으로 활발한 연구가 시작되어 1995년경 흔TEX으로 일단 완성을 보았다. 흔TEX이 훌륭한 프로그램이었음에도 불구하고 시장화에 실패하여 사장된 후, 흔TEX과 같은 맥락에서 만들어졌던 차재춘의 hLATEXp와, 그와는 조금 다르게 독자적인 폰트 디자인을 채택하였던 은광희의 HLATEX이 사실상 한글 LATEX의 중심을 이루어 왔다. 이 둘은 모두 당시 이른바 "완성형 한글" 코드—KS C 5601, 나중에 KS X 1001의 일부—를 사용하여 한글을 식자하던 LATEX 매크로였다.

ChoF's TeX Archive와 도은이네 집에서는 HLATEX을 중심으로 사용자에 대한 지원과 이런저런 질문 답변을 행해왔다. 특히 ChoF's TeX Archive의 몇 가지 팁 가운데 hLATEXp와 HLATEX의 호환성을 해결할 수 있는 몇 가지 힌트가 주어져서 HLATEX의 활용성을 높인 점이 특기할 만하다.

2001년에 KTUG이 만들어지면서 KTUG의 대부분의 활동은 HLATEX을 중심으로 이루어지게 된다. KTUG이 HLATEX에 주목했던 이유는 대략 다음과 같다.

(1) hLATEXp가 일부 그룹에서 여전히 쓰였지만 1996년 이후 사실상 업데이트와 지원이 중단되었다. 상업화에 실패한 후 버려졌다는 표현이 아마 맞을 것이다. 반면 HLATEX은 꾸준히 새로운 버전이 나오고 있었다.

(2) hLaTeXp의 폰트 문제는 치명적인 것으로 보였다. 특히 최종 출력 포맷을 PostScript 폰트를 내장한 ps 또는 pdf로 하는 문제에 직면해 있던 당시로서 pk 폰트 파일을 제공하는 hLaTeXp의 한계는[1] 품위 면에서 참을 수 없을 지경이었던 반면, HLaTeX은 폰트 디자인 자체에 대한 호오를 논외로 하더라도 PostScript type 1 한글 폰트를 제공하였다는 것이 대단히 매력적으로 보였던 것이다. 이 폰트 자체를 디자인한 것이 실로 HLaTeX의 가장 큰 업적이었다고 할 수 있으며, 아직도 우리는 그 혜택을 받고 있는 셈이다.

2.2 문제

KTUG이 만들어진 첫 모임에서 논의되었던 것을 중심으로 당시 인식되고 있었던 문제점을 요약하면 다음과 같다.[2]

"완성형 한계"의 문제 유명한 '똠방각하'의 표기뿐 아니라, 2350글자 완성형만을 식자할 수 있다는 한계는 실용적으로 볼 때 큰 문제거리가 아니라 해도 해결해야 될 것임에 틀림없었다.

PDF 제작 문제 dvi를 중심으로 하는 시대가 지나고 pdf라는 새로운 표준에 알맞도록 한글 pdf를 보다 효과적으로, 고품위로 제작할 수 있도록 하자는 데 의견이 모아졌다. 적어도 hLaTeXp는 당장 폰트 때문에라도 이 요구에 대응할 수 없었고, HLaTeX은 폰트 문제는 어느 정도 해결되었다 할 수 있으나, 한글 텍스트의 검색 및 추출, 한글 pdf 북마크, 하이퍼링크 등의 문제가 해결되어야 할 것으로 떠올랐다.

폰트 문제 한글 트루타입 폰트를 사용할 수 없을 것인가가 화제의 중심에 떠올랐다. 실제 조진환은 개인적인 실험의 수준이기는 하였으나 윈도 바탕 글꼴을 이용하여 TeX 식자에 성공한 경험을 가지고 있었으며, 한글 폰트가 당시 트루타입 중심으로 제작되고 있던 점에 주목하였던 것이다. 이 문제는 KTUG의 발족 직전 박원규, 조진환, 김도현 등에 의하여 상당한 논의의 진척을 보이고 있던 분야이기도 하였다.

---

1) hLaTeXp를 제작한 곳에서는 mf 폰트 원본을 사용할 수 있었다고 한다. 그러나 일반 사용자에게는 pk 폰트만이 주어졌다.

2) 초창기 KTUG의 활동은 많은 부분 사용자에 대한 설치 지원에 할애되었다. 지금 돌이켜보면 이것은 고급 개발 인력을 낭비한 것은 아니었던가 하는 생각이 들기도 하는데, 이를 통하여 TeX의 대중화가 촉진된 것은 업적으로 기록해두어도 부끄럽지 않을 것이다.

3 HLaTeX을 통한 실험과 개선 : 2002–2004

3.1 한글 트루타입 폰트 사용하기

트루타입 폰트를 사용하기 위해서 해결되어야 할 문제는 많았다. 여기서 "트루타입 사용"이란, 적어도 트루타입의 윤곽선 정보를 최종 출력물 pdf에 넣을 수 있을 것을 의미한다. 즉, pk 글꼴로 변환하여 처리하는 방식을 의미하지 않는 것이다.[3)]

1. 트루타입 폰트 자체가 있어야 한다. 당시에는 공개된 한글 트루타입 글꼴이 없었기 때문에 윈도 기본 글꼴과 무료 공개하던 아시아 폰트를 위주로 개발이 이루어졌다.
2. 트루타입 글꼴을 TeX 글꼴로 변환해야 한다. 이 작업은 처음에 조진환에 의하여 부분적으로 이루어졌다. 실제로 HLaTeX 0.98에는 문화부 트루타입이 포함되어 있었으므로, 적어도 pk 글꼴로 변환하여 트루타입을 사용하는 방법은 이미 HLaTeX 자체에 구현되어 있었던 셈이다. 그러나 우리가 원한 것은 pdf를 윤곽선 글꼴로 만드는 것이었지 pk 비트맵으로 변환하여 처리하는 것이 아니었다.
3. 트루타입 글꼴이 포함된 dvi를 처리할 dvi 드라이버가 요구되었다. pdf 변환을 의도하고 있었기 때문에 dvipdfm과 pdfTeX이 집중적으로 검토되었다.

이 문제들은 하나하나 착실히 해결되어 갔는데, 대략 그 경과를 요약하면 다음과 같다.

(1) 트루타입 폰트는 '은 글꼴'의 탄생으로 극적인 전기를 맞았다. TeX 사용자는 물론이고 그 외의 영역에 있어서도 GPL 라이센스를 가진 자유 글꼴 한 벌이 생겨난 것은 중요한 변화라고 해야 할 것이다. 이 글꼴은 원안을 디자인한 은광희, 트루타입으로 변환한 박원규, 신정식 등의 노력의 결실로서, 특별히 기록하여 둘 가치가 충분하다.

(2) 처음에 pdfTeX을 중심으로 시작되었던 트루타입 처리 방식에 관한 논의는, 마침내 조진환에 의하여 dvipdfmx-cjk를 거쳐 DVIPDFM*x*에 이르러 해결을 보게 된다. DVIPDFM*x*는 sub-font로 작성된 dvi로부터 원래의 트루타입을 읽어 pdf에 내장(embed)해주는 기능을 갖추고 있었다. 물론 설정에 따라 폰트를 embed하지 않도록 설정할 수도 있었다. 이것은 대단히 획기적인 발전으로, 이 당시 트루타입 한글 폰트를 이용하여 검색 가능한 pdf를 제작하고 pdf 북마크를 붙일 수 있는 도구는 DVIPDFM*x*가 유일했다. 이 사정은 최근에 이르기까지 별로 나아지지 않았으나, 최근에는 pdfTeX 등을 이용한 방법도 사용이 가능해졌다.

3) 이와는 별도로, dvi viewer를 사용할 때는 여전히 트루타입으로부터 pk 글꼴을 생성해낼 수 있어야 한다.

이 두 가지 근본 문제가 해결됨으로써, TEX 사용자는 비교적 자유롭게 글꼴을 선택할 수 있게 되었다. 김도현의 `ttf2hlatexfont` 스크립트가 제작되어 글꼴 사용을 도와주게 되었으며, 아시아폰트 패키지와 같이 미리 정의된 폰트 패키지도 제작하여 배포하게 되었다. 이 성과는 놀라운 바가 있어, 그 후로 pdf 제작은 DVIPDFM*x*로 하는 것이 바람직한 것으로 권장된 것은 우연이 아닐 것이다.

3.2 고품위 pdf 제작

트루타입 문제가 해결되자, 곧 고품위 pdf 제작 문제가 본격적으로 논의되기 시작했다. 고품위 pdf라고 하면 여러 의미가 있겠지만, 당시로서 논의의 중심에 있었던 것은 다음 몇 가지 조건을 갖추는 것이었다.

1. 윤곽선 글꼴로 제작된 pdf.
2. pdf bookmark와 같은 도구.
3. 하이퍼링크.
4. 한글 텍스트의 검색과 추출.

이 가운데 항목 1, 2 및 4에 해당하는 문제가 DVIPDFM*x*로 해결이 된 것이다.[4] 즉, DVIPDFM*x*는 트루타입 한글 글꼴의 사용뿐 아니라, 한글 pdf bookmark와 텍스트 검색·추출도 가능하게 하였던 것이다.

이제 문제는 남은 한 가지, 즉 하이퍼링크를 어떻게 구현할 것이야 하는 점이었다. 당연히 이것은 `hyperref` 패키지를 써야 했지만, 문제는 HLATEX의 `hangul`이라는 막강한 영향력을 가진 패키지가 `hyperref`과 충돌한다는 것이었다. 그것도 자동조사라는 치명적인 기능을 사용할 때.[5]

김강수는, `hangul`에서 자동조사 기능을 제외하여 `hangul-nojosa`라는 패키지를 만들었다. 자동조사 없는 한글 패키지란 있을 수 없는 것임을 숙지하고 있었으나, 이 기능을 잠시 잠정적으로 제외한다면 적어도 pdf 하이퍼링크를 사용할 수 있었던 것이다. 이것은 말 그대로 잠정적 조치였고, `hyperref`과 자동조사 문제를 해결하는 것은 한시가 급한 일이었다.

4) pdfTEX이나 dvips에서 같은 기능을 어떻게 구현할 것인가는 일단 후일의 문제로 남겨졌다.

5) HLATEX 0.991까지만 해도 hangul 패키지를 hyperref과 함께 쓰면 컴파일이 되지 않았다. 1.0.1 버전에서 khyper를 함께 사용할 때 컴파일이 되지 않던 문제는 발생하지 않으나 hyperref의 기능들을 활용할 수는 없다.

이 문제를 해결한 것은 김도현이다. 그는 hyperref이 만들어내는 링크를 역으로 읽어, 거기에 맞게 조사를 붙여주는 루틴을 새로이 제시하였다. 이것으로 hangul의 자동조사를 대신하도록 한 것이 hangul-k라는 패키지였는데, 김도현과 김강수의 협력의 결과 공동 명의로 발표된 이 패키지로 인하여 마침내 HLaTeX과 hyperref의 충돌 문제를 거의 완전하게 해결할 수 있었다. 문서작성자가 사용하는 매크로 명칭은 동일하게 하였기 때문에 hangul-k는 (내부적인 루틴은 상당히 달랐지만) hangul과 똑같이 작동하면서도 hyperref과 충돌하지 않게 만들 수 있었던 것이다.[6] 필자는 이 "새로운 자동조사 루틴"을 KTUG이 얻어낸 매우 중요한 성과 중의 하나라고 생각한다.[7]

3.3 완성형 밖의 한글 및 옛한글 표현하기

"똄방각하"나 "커피숖"과 같은 EUC-KR로 표현할 수 없는 한글의 식자 문제는 신속히 해결하지 않으면 안될 긴급한 문제는 아니었지만[8] 당시 이미 CP949로 모든 한글을 자유롭게 표현하고 있었던 Windows 운영체계에 비추어볼 때 좀 시대에 뒤떨어진 느낌을 주는, 일종의 자존심 문제였다.[9]

또다른 한 가지는 소위 '고어' 즉 옛한글의 표현 문제였다. 예컨대 워드 프로세서 흔글의 이름을 문서에 적어야 할 때 아래아를 식자할 방법을 찾지 못하고 있었던 것이다.[10]

이 두 가지는 서로 밀접히 연관된 문제로, 한글 코드 문제에서부터 시작해서 폰트 문제에 이르기까지 수많은 문제가 얽혀 있는 것이었다.

우선, HLaTeX 자체가 LaTeX이 아니라 Lambda로 실행하면 UHC(통합한글코드) 영역의 모든 한글 11,172자를 표현할 수 있다는 사실은 일찍부터 알려져 있었다. 그러나 호환이나 폰트 설정에 여전히 문제를 안고 있던 Omega를, 단 한 글자를 식자하기 위해 실행한다는 것은 그것이 가능하다 하더라도 경제적이지는 못했다고 생각한다.

---

6) 이 글을 쓰는 현재 hangul-k 패키지를 사용하는 것이 불가능하지는 않다. 그러나 EUC-KR 한글 패키지에 대한 지원은 현재 완전히 중단된 상태이므로(HLaTeX을 제외하면) hangul-k를 사용하는 것은 그다지 권장할 만한 것은 아니다. hangul-k의 아이디어는 dhucs(a.k.a. hangul-ucs)에 계승되었다.

7) 여담이지만, "hangul-k 매뉴얼"이라는 문서는 필자가 작성한 문서 중 가장 자랑스러워하는 것 중 하나이다. 현재는 시의적절하지 않으나 당시로서는 매우 진보적인(!) 내용을 담고 있었다.

8) 다만, 예외적으로 이호재는 "띡"이라는 의성어(tick)를 문서에 사용하고 싶어 했는데, 이 글자가 표현이 되지 않은 경험을 말한 바가 있다.

9) 현재도 WinEdt 사용자들은 이 글자들을 표현할 방법을 알지 못한다. 게다가 오늘날의 윈도 사용자들은 EUC-KR 문자, 또는 완성형 문자가 무엇이고 왜 문제가 되는지 이해하지 못한다. 자연히, LaTeX이 뭔가 문제가 있는 프로그램인 것처럼 생각하게 되기 쉬운 것이다.

10) 흔글을 '표시'하는 재미있는 아이디어들도 제법 제출되었다. 몇 가지 들어보면, 제일 단순한 것이 이것을 그림으로 처리하는 것이었고, 다른 예로는 ㅎ과 받침 ㄴ을 박스로 만들어 가운데 점을 하나 추가하여 찍는 '기발한' 방법도 있었다. 그러나 그것은 엄밀한 의미에서 흔글을 식자하였다고 할 수는 없는 것들이었다.

LaTeX을 여전히 사용하면서 모든 한글을 표현하는 방법을 모색하던 중, CJKLaTeX이라는 Werner Lemberg의 패키지가 주목을 받았다. 이 패키지는 한글 유니코드(UTF-8) 입력을 처리할 수 있는 능력이 있었던 것이다. 일단 한글 코드를 유니코드로 전환하자, 모든 한글은 물론이고 (비록 한양 글꼴에 종속적이기는 하였지만) 옛한글도 표현할 수 있다는 사실이 알려지게 되었다. 그러나 CJKLaTeX은 HLaTeX의 중요한 기능들, 예컨대 자동조사와 같은 한국어에 밀착한 기능들이 결여되어 있어 널리 사용하기에는 한계가 있었고, 몇 가지 테스트 수준에 머무르게 되었다.

김강수가 재미삼아 만들었던 `hlatexcjk`라는 패키지는 옛한글 몇 글자를 식자할 필요가 있을 때 CJKLaTeX의 `UTF8` 환경을 빌려쓰자는 발상에서 나온 것이었다. 해결책(solution)이라고 하기에는 뭣하지만, HLaTeX을 여전히 쓰면서도 그 한계를 넘어서려 했던 기록으로 여기 적어둔다.

완성형 밖의 글자를 표현하는 방법을 천착하면서, 많은 트루타입이 실제로 완성형에 해당하는 글리프밖에 갖추고 있지 않음을 확인하게 된 점도 적어두어야 할 것이다. 현대 한글 음절 문자를 전부 포함하고 있는 글꼴은 은 글꼴, 윈도 기본 글꼴 정도뿐이었으며, 아시아 글꼴 등은 완성형 이외의 문자에 해당하는 글리프 자체가 없었다. 이 사정은 지금도 나아지지 않아서, 실제 판매되는 한글 폰트의 대부분이 완성형 자소만을 갖추고 있다. 그런 점에서 은 글꼴이 처음부터 11,172자의 모든 현대 한글 음절 문자를 다 포함하고 있었던 점은 행운이라고 해야 할 것이다.

3.4 Omega를 이용한 실험

은광희는 이 과정에서 (신정식의 도움을 받아) UTF-8 한글을 다루는 새로운 HLaTeX의 Lambda 판을 발표하고 이를 `u8hangul`이라 불렀는데, 이 스타일은 2005년의 1.0 버전 Lambda 부분의 토대가 되었다.

옛한글 문제는 2002년의 KTUG을 뜨겁게 달군 주제였다. 이 문제에 관한 한, Omega를 중심으로 한 논의가 심각하게 이루어졌다.

1. 조진환은 CJKLaTeX과 한양 새바탕 또는 새굴림 글꼴을 이용하면 옛한글을 식자할 수 있음을 처음으로 보여주었다.

2. 조진환은 한양 옛한글 자모 서체인 한양 옛바탕 글꼴을 이용하여 자모를 조합하는 규칙을 발견하고 이를 Omega에서 사용할 수 있도록 `otp`로 구현하였고, 신정식 등이 이 규칙을 개선하여 mslvt라는 Lambda 패키지가 만들어졌다. 이 패키지는 `u8hangul`에서 동작하는 것이었다.

3. 옛한글의 표현이 가능해지자, 자연히 한양 PUA 영역의 코드에 대한 비판적 논의가 시작되었고, 곧 유니코드 한글 자모 영역을 이용하는 소위 '첫가끝' 코드를 표준으로 받아들이는 쪽으로 논의가 자연스럽게 진행되어 갔다.[11)]

4. 김도현은 이러한 모든 논의를 집대성하여 dhhangul이라는 Lambda 패키지를 제작하였다.

3.5 dhhangul

김도현의 dhhangul은 아마도 Omega에 관하여 이루어진 KTUG의 논의의 최종판일 것이다.[12)] 상당히 많은 테스트가 옛한글과 고문헌의 식자에 대하여 이루어졌고[13)] 이 과정에서 dhhangul 자체도 많은 변화를 겪어 갔다. 예컨대 dhhangul은 코드 영역에 따라서 글꼴을 분리하는 정책을 채택하여, 옛한글 영역, 한자 영역, 한글 영역, 라틴 문자 영역에 모두 다른 글꼴을 배당할 수 있게 하고 있었다. 이런 정책은 사실은 이 모든 영역을 모두 아우를 만한 폰트가 없었다는 것이 가장 큰 이유였는데, 초창기에는 한글을 제외한 한자 영역에는 `cyberbit`을 적용하다가, 나중에 글꼴 자체를 선택할 수 있는 쪽으로 발전하기도 하였다. 다만, Omega 자체가 가진 시스템의 불안정성은 늘 우리를 불안하게 하였다. Omega로는 pdf 북마크를 결국 만들 수 없었던 것 같은 것이다. 훗날 마침내 Omega가 개발 중단에 이르게 되어, 이 모든 논의가 사실상 효용이 생각만큼 크지 않은 업적으로 남게 되었다. 유감스러운 일이나, 우리로서는 이 토론의 과정에서 얻은 것이 아주 많고 그것은 LaTeX으로 되돌아왔을 때도 실로 많은 도움을 주었다.

여기까지 이르러서, 한글 LaTeX 또는 Omega/Lambda로, 적어도 한글에 관한 한 어떤 문자든지 식자할 수 있다는 자신감을 가지기에 이른다. 충분한 품위의 글꼴만 있다면.

은 글꼴이 훌륭하나, 한자 영역은 EUC-KR의 4888자를 벗어나지 못하고 있었고(이것으로는 옛문헌을 도저히 식자할 수 없다), 옛한글은 박원규에 의하여 은 바탕 글꼴에 대해서만 테스트되고 있던 형편이었다. 현재 옛한글을 표현할 수 있는 모든 폰트가 전부 상업용이라는 점은 이 문제의 심각성이 어느 정도인지를 보여준다.

---

11) 그러나 한양 PUA를 버릴 수도 없는 일이라 이를 활용하기 위한 HanyangPuaTable의 작성이 이로부터 얼마 후에 이루어지게 된다. 이 프로젝트에는 많은 사람들이 참여하여 단기간에 괄목할 성과를 내게 되었다. 이제는 자모 코드와 한양 PUA의 상호 변환이 매우 용이해졌는데, 이것은 이 프로젝트의 성과에 바탕을 두고 있다.

12) 이와는 별도로 다국어 조판에 관련된 조진환의 Omega-CJK라는 패키지도 작성된 바가 있다. 이 패키지는 2004년 TUG Conference에서 발표되었다.

13) TeX으로 조판된 《논어》, 《대학》을 볼 수 있었던 것은 이 때 와서 처음이었다. dhhangul 패키지로 논어집주 조판에 성공했을 때의 감동이 생생하다.

3.6 대안적인 TEX 한글 구현

이 당시부터 ConTEXt라는 새로운 TEX 매크로에 대한 관심이 커졌다. 그 주된 이유는 역시 LATEX이라는 도구로 할 수 없는 ConTEXt 특유의 막강한 기능이 가장 매력적인 것이었다고 할 수 있고 HLATEX의 문제점—주로 다른 패키지와의 충돌—을 벗어날 수 있는 기회가 되지 않겠는가 하는 막연한 기대가 작용한 것도 사실이다.

조진환이 초창기 ConTEXt 한글화 매크로를 시험삼아 작성한 것이 있었으며, 이것은 아마 ConTEXt에서 한글을 사용할 수 있게 된 최초의 시도였을 것이다.

plainTEX은 주된 관심의 대상이었던 적은 없으나 조희대가 HLATEX의 uhc 폰트를 사용하여 EUC-KR 한글을 plainTEX에서 표시할 수 있음을 보여준 바가 있다.

4 유니코드 한글 : 2005–2006

4.1 한글 코드 문제

유니코드 사용이 한글 식자의 일반적 문제를 근본적으로 해결할 수 있는 길임을 깨달은 것은 Omega와 Lambda를 집중적으로 테스트하면서부터였다.

한글 표현 방법에 있어서 EUC-KR의 2350자는 낡은 것이 되어 갔다. 그 대안은 둘밖에 없었는데 하나는 유니코드이고 다른 하나는 이른바 "확장완성형"이라 불린 Windows CP949 코드였다. Omega/Lambda를 문제삼는다면 이 가운데 어떤 코드를 채용해도 해결책이 있었겠지만 8비트씩 처리해야 하는 LATEX에서는 현실적으로 CP949를 채택할 수 없는 사정이 있었다.[14] 게다가 CP949는 표준도 아니며(윈도를 제외하면) 리눅스나 맥 어떤 다른 시스템에서도 이런 코드를 채택하지는 않았으므로 이 문제에 있어서도 선택의 여지는 없었다.

한글 코드가 '문제'인 이유는 주로 윈도라는 운영체제 때문이다. 일반적인 윈도 사용자들에게 유니코드는 낯선 것일 수밖에 없었고, 더욱이 WinEdt이라는 에디터가 유니코드를 전혀 지원하지 않음으로써[15] 문제를 더 복잡하게 만들었다. 한글 코드 문제가 어찌되든 적어도 TEX에서 현재는 유니코드 이외에 대안이 없고, 따라서 이것을 제대로 지원하지 못하는 WinEdt은 포기하는 도리밖에 없는 것이다.

---

14) 그 사정 중의 하나는, CP949 16비트를 8비트씩 쪼개었을 때 TEX의 유보문자에 해당하는 코드가 나올 수 있다는 것이었다. 그 가운데서도 특히 category code 2와 3에 해당하는 {, } 문자가 출현할 때는 상당히 골치아픈 문제를 야기했다.

15) 한때 WinEdt 도움말에 "유니코드는 inflated ascii에 불과하다"는 표현이 나온 적이 있다. 이것은 유니코드의 필요성을 악센트 붙은 영문자를 표현하는 정도로 이해하고 있었다는 뜻이다.

운영체제로서 윈도 자체는 유니코드를 매우 잘 다룬다. 특히 Windows 2000 이후로는 내부적으로 유니코드를 이용하기 때문에 유니코드 문서 등을 작성하는 에디터도 훨씬 많고 편리해졌다. 그러나 사용자 인터페이스는 여전히 한글의 경우 CP949라서 처음사용자들이 혼란스러워하는 경우도 있다. 이를 극복하는 방법은 아마도 잘 설정된 편리한 에디터를 통하는 방법밖에 없지 않을까 싶다.

4.2 hangul-ucs의 등장과 발전

LaTeX에서 유니코드 사용은 Dominique Unruh의 ucs 패키지가 기초를 마련해두었다. 이것은 현재 LaTeX에서 "사실상 표준"으로 받아들여지고 있는 유니코드 처리 패키지가 되었다. 김도현의 dhucs는 이 ucs 패키지의 유니코드 처리 능력에 한글 문서가 요구하는 한글화 기능과 설정을 추가로 구현하여 만들어졌다. 이제 LaTeX에서도 유니코드 한글을 처리할 수 있게 된 것이다.[16)]

이 패키지(hangul-ucs)는 사실상 HLaTeX의 강력한 대안으로 성장해갔다. 처음부터 은 글꼴을 기본 글꼴로 채택함으로써 DVIPDFM*x*를 이용하여 고품위 pdf를 제작하는 것을 중점적으로 지원하였으며, dhucs에서 구현된 한글 처리 기능을 LaTeX뿐 아니라 ConTeXt, JadeTeX, plainTeX 등으로 확장해갔다.[17)] 현재 LaTeX을 비롯한 TeX-류에서 한글을 완전하게 지원하는 모든 방법은 dhucs에서 비롯된 것이다.

dhucs는 종래 제기되어 온 LaTeX에서 한글 사용 및 pdf 제작의 거의 모든 문제들에 대한 해법을 제시하기도 한다. 이것을 요약하면 다음과 같다.

1. 한글 표현의 제약이 없어졌다.
2. 고품위 pdf 제작을 기본적으로 지원한다. 즉, 한글 북마크, 텍스트의 검색 및 추출, 윤곽선 글꼴의 사용 등이 hangul-ucs와 DVIPDFM*x*의 결합으로 전부 해결을 보게 되었다.
3. 폰트만 지원된다면 옛한글이나 고문헌의 식자도 Lambda를 빌리지 않고 LaTeX으로 가능하게 되었다.
4. 다국어 문서를 작성할 수 있게 되었다.

16) 리눅스 사용자들에게 hangul-ucs가 친숙해진 데는 이호석 님이 제작한 데비안 패키지의 공이 결정적이었다고 하지 않을 수 없다. 박원규 님도 `rpm` 패키지를 제작하였다. 그 이전까지는 hangul-ucs가 개발되고 있다는 사실 자체가 알려져 있지 않은 상황이었다. hangul-ucs의 개발에 참여하거나 지켜본 입장에서, 잘 만들어진 설치 패키지 하나가 미치는 강력한 영향에 놀란 바 있다.

17) Windows 용 KC2006(W32TeX/ko)은 pdfTeX으로 `pdf`를 제작하는 데도 아무런 어려움이 없으나, 좀 낮은 버전이 일반화되어 있는 리눅스나 매킨토시 시스템에서는 이 분야에 있어 조금 시간이 걸릴 수도 있으리라 짐작한다. 표준 배포판의 pdfTeX이 1.40 이상 버전이 되면 편리할 것이다.

5. 한글 타이포그래피의 구현에 대해 이루어진 성과를 대부분 포함하고 있다. 그 가운데는 HLaTeX의 "나쁜 행나눔"을 개선한 것이 가장 중요하고 자간설정, 금칙처리, 방점, non french spacing 등 문서 작성에 필요한 대부분의 기능을 구현하거나 개선하였다.

냉정하게 말하자면, 인쇄물을 만들어도 좋을 정도의 한글 문서를 작성할 수 있게 된 것은 dhucs, 즉 hangul-ucs 발전 이후의 일이었다고 해도 좋을 것이다. 그 이전에는 한글이 찍히는 것 자체가 신기한 수준을 넘어서지 못하였다. 이 성과가 별것아닌 것으로 치부될 수 있을는지는 모르나, 적어도 한글 문서의 작성과 출판이라는 관점에서 상당한 진보를 이루었다고 스스로 평가하고 있다. 이것은 종래의 한글 LaTeX이란 것이 만든 판면과 최근의 출력물을 직접 비교해보면 한눈에 알 수 있는 것이리라.

5 KTUG Collection 2006의 제작

KTUG 창립 초기부터 논의가 진행되던 "표준 한글 TeX 환경 프로젝트"[18]는 2005년에 이르러 마침내 KTUG Collection을 제작하게 되었다. 이 때의 중점적인 요인은 그 당시까지 개발되었던 한글 관련 모든 솔루션을 합쳐서 하나의 단일한 texmf tree로 제공한다는 것이었다. 이를 바탕으로 윈도에서 설치 가능한 CD를 MiKTeX 기반으로 제작하여 배포한 바 있다. 이 collection의 중심은 HLaTeX 0.991과 dhucs였다.

소기의 성과는 거두었다고 우리는 스스로 평가하고 있기는 하나, HLaTeX 1.0.1이 KC-2005와 거의 동시에 버전업됨으로써, 사실상 KTUG Collection은 빛을 잃은 면이 있다. 이 당시까지는 dhucs를 강제하거나 중시할 형편이 아니었고, KTUG Collection 2005 자체는 HLaTeX 0.991을 채택하였는데, 하필이면 새로운 버전이 며칠을 사이에 두고 새로 릴리스된 것은 KC2005의 운명이라 해야 할 것이다.

이 경험을 바탕으로 하되, KTUG Collection 2006은 이와는 다른 배경에서 출발하였다. 즉, MiKTeX을 포기하고, W32TeX/ko라는 새로운 시스템을 받아들인 것이다. 이 시도는 아직 그 공과가 확연히 드러나지는 않았지만 W32TeX 자체가 워낙 훌륭한 시스템이므로 성능에 있어서 문제는 있을 수 없다고 생각한다. 거기에다가 KC2006은 다양한 사용자 지원 수준의 유틸리티들과, 한글 문서 작성에서 느끼는 각종 애로를 모두 반영한 세밀한 설정으로, 현재 사실상 한글 문서 작성의 표준을 바라보고 있다 하겠다. KC2006을 둘러싸고 KTUG에서는 격렬하다면 격렬한 논쟁이 진행되기도 했는데, 주된 논점은

18) 홍석호와 필자가 참가하였다.

MiKTEX이라는 편리한 환경을 두고 KC2006으로 넘어가야 할 당위성이 부족하다는 데 초점이 맞추어졌다. 지금 이 논쟁을 생각하면, MiKTEX에서 편하게 한글 LaTeX 작업을 할 수 있었던 자체가 KC2005의 성과였음을 인정하지 못하거나 의식하지 못하는, 논점이 잘못 설정된 논란이었다는 생각이 든다. 게다가 한글을 주로 사용하지 않겠다는 분과는 논쟁이 성립하지 않는 것이었다.

몇 가지 문제점[19] 때문에 리눅스 및 매킨토시 버전을 즉시 제공하지 못하는 한계에도 불구하고, 지금까지 윈도 버전이 리눅스보다 더 진보한 시스템인 경우가 별로 없었으므로 그 점에서도 KC2006은 특기할 만한 점이 있다고 생각한다. 리눅스와 매킨토시 버전의 KC2006에 대해서는 현재 논의가 진행중이다.

6 결어: 무엇이 달라진 것인가?

대략 지금까지의 논의를 요약해보자.

(1) 사실상 `pdf`를 표준 출력으로 받아들이게 되었다.

(2) 한글의 타이포그래피에 획기적 진전을 이루었다. 종래 HLaTeX이나 hLaTeXp는 한글 타이포그래피에 관한 한 사실상 아무런 조치도 취하지 않은 것이나 다름없다는 주장을 감히 해도 될 것이다.[20]

(3) LaTeX 이외의 다른 매크로에 대해서 한글 지원이 이루어지게 되었다.

(4) 특히 pdf에서 한글 텍스트의 검색과 추출, 그리고 한글 북마크를 완벽하게 지원하게 된 것은 매우 중요한 일이라고 생각한다.

(5) 유니코드 한글을 일관되게 사용하게 되었다. 즉, 옛한글을 포함하여 모든 한글을 전부 표시할 수 있게 되었다.[21]

출판 품위와 관련해서 `memoir`의 한글화를 지적하지 않을 수 없다. `memoir` 그 자체는 하나의 도큐멘트 클래스에 불과하지만, 이를 통하여 TEX 문헌의 출판 가능성이 열린 점은 매우 중요하다. 즉, 한글로도 `memoir`를 사용할 수 있게 되었고, 이러한 클래스 또는

19) 예를 들면, KC2006의 pdfTEX이 1.40.x라는 점을 들 수 있다.

20) HLaTeX 1.0.1의 기능은 자간이나 행간, 각주 모양과 같은 많은 점에서 상당한 진보를 이룬 바가 있지만, 이 글에서 비교 대상은 HLaTeX 0.991이다.

21) 텍스트의 검색과 추출까지 모두 가능하다. 그러나 소위 '첫가끝' 코드에 대한 토론은 여전히 유효하다고 생각한다.

패키지의 사용에 아무런 문제를 느끼지 않게 되었다는 점은 KTUG 5년의 개발 과정이 헛되지 않았다는 것을 보여준다고 하겠다.

7 結

사실상 한글 LaTeX 사용 환경은 현재 거의 완성되어 있다고 해도 과언이 아니다. 이만큼의 결과를 얻기까지 실제로 한글 LaTeX 개발에 헌신적으로 기여해주신 분들, 은광희, 조진환, 김도현, 신정식, 박원규, 이주호, 김병룡, 조희대 諸氏에게 감사를 드리지 않을 수 없다. 이 분들의 헌신적인 노고와 기여에 의해, 오늘 우리는 LaTeX에서 한글을 마음껏 사용하는 자유를 누리고 있는 것이다. 이름을 열거하지 않은 많은 분들의 기여도 결코 가볍다 할 수 없을 것이다. 이름을 누락한 것은 그분들의 기여가 가볍지 않아서가 아니라, 다만 필자의 기억력의 한계일 뿐이다. 또한, 이름을 열거할 수조차 없는 많은 분들의 실제 사용과 질문 답변은 한글 LaTeX의 발전에 실질적인 추동력이 되었다. KTUG 사이트가 정말로 가치를 갖는 것은 그런 점에서일 것이다.