TEX과 그 언저리

Karnes*

2006년 9월

요 약

이 글은 2006년 9월 17일 KDLP F/OSS 컨퍼런스의 부대 행사로 준비하는 KTUG BoF의 토론 자료로 작성하는 것이다. TEX에 관심이 있거나 언젠가 한번쯤 TEX을 써보았던 기억이 있는 분들에게 최근 TEX 시스템 자체의 변화와 발전을 개략적으로 소개하고 토론을 유도해내는 것이 목적이기 때문에 자세한 보고서 형식의 글이 되지는 못하였다.

〈 목 차 〉

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

1 緖

이 짧은 개요는 TEX과 그 주변 관련 프로그램 및 매크로들의 현재 상황과 TEX의 이용 현황 및 앞으로의 발전 방향에 대한 논의들을 일별하려는 목적으로 쓰여진다. 이런 종류의 글을 쓰기에 필자가 적합한 것 같지는 않지만, 혹시라도 도움이 될는가 하여 메모 수준의 정리를 행하는 만용을 저지르게 되었다. 주마간산 격의 내용없는 요약이라도 용서해주시기 바란다.

1.1 용어

TEX이라는 용어에 관련해서 몇 가지를 미리 적어두려 한다. 다른 곳의 발표[1]에서 언급한 것이기도 한데, TEX은 적어도 몇 가지 다의적 의미를 지니고 있다.

(1) 문서 조판 언어(language). 나름의 문법과 구조를 갖춘 조판 프로그래밍 언어라는 의미로 쓰일 때의 'TEX'이다.

(2) 소스 컴파일러 프로그램. 이것은 `.tex`이라는 확장명을 갖고 사용자가 조판 사항을 지시한 원본(소스) 파일을 `.dvi`라는 표준 출력으로 변환해주는 프로그램을 가리키는 말로 쓰인다. 이 프로그램은 명령행에서 `tex`이라고 부름으로써 실행시킬 수 있다.

(3) 매크로 패키지. TEX language로 쓰여진 매크로 집합으로서, 실제의 문서 작성을 도와주는 여러 매크로의 사전 정의들로 이루어져 있다. 가장 유명한 두 개의 매크로 패키지를 들라면 Knuth 자신이 작성한 plainTEX과, LATEX이 있는데, 이들은 이를테면 실제 문서작성을 위해 필요한 명령들을 미리 정의해둔 라이브러리에 해당한다.

통상 "TEX으로 문서를 작성한다"고 하면 LATEX을 가리키는 경우가 많고, "LATEX이 아니고 TEX"이라고 하면 plainTEX을 지칭하는 것인데, 이 글에서는 이들을 일관되게 "매크로 패키지"라고 불러서 다른 두 가지와 구별했다. 앞으로 명백해지겠지만, 오늘날 "TEX"이라 함은 실제로 "TEX 확장" 프로그램과 언어를 총칭하는 말로 쓰이는 경우도 많다.

"TEX을 확장한다"함은, 매크로 패키지(예컨대 LATEX)에 새로운 매크로를 추가하는 것을 의미하는 것이 아니라, `tex` 프로그램의 소스[2] 수준에서 새로운 primitive를

[1] 수원대학교 2006. 5.

[2] `WEB` 또는 Web2C에 의하여 변환된 `C`가 흔히 쓰인다.

추가하거나 변경하여 TEX 언어의 설계와 기능을 바꾸거나, 함수의 동작을 수정하거나 새로운 함수를 추가하여 `tex` 프로그램의 동작 자체를 개선하는 것을 의미하는 것으로 한다.

2 TEX

TEX의 현재 버전은 `3.141592`이다. 소수점 아래 여섯 번째 자리 업그레이드는 2002년 12월에 이루어졌고, 모두 네 가지 버그를 수정한 것이라 한다. 그 중 두 가지는 `leaders`에 대한 것이고, 나머지 두 가지는 정렬(alignment)에 관한 것이다. 이와 함께 METAFONT도 2.71828로 버전업이 이루어졌다.[3] 한때 `3.14159`에서 프로즌 되었다는 얘기가 있었지만 그것은 사실이 아니고, 앞으로도 버그 수정은 계속될 것이라고 보아야 한다(만약 버그가 남아 있다면). 물론 TEX 자체의 디자인은 모두 종료되었다고 할 수 있다.

도대체 TEX이란 무엇인가? METAFONT라는 "이제는 낡은" 폰트 처리 기술, 이제는 일반화되어 그다지 새로울 것도 없는 개행 알고리듬, 그리고 아무도 사용하지 않는 Pascal-`WEB` 언어, 여전히 8비트에 묶여 있는 코드 처리 방식에 따른 유니코드 처리의 한계, 게다가 복잡한 문서에서 종종 맞닥뜨리지 않을 수 없는 255(256)개의 register 제한 때문에 벌어지는 "`TeX capacity exceeded...`" 메시지…. 오리지널 TEX은 아무리 보아도 오늘날 비영어권에 살면서 문서를 만들어야 하는 우리의 요구를 충족하지 못한다. 그리하여, (여전히 우리가 TEX이라고 부르고는 있지만 사실은 TEX이 아닌) 여러 종류의 확장된 TEX을 통해서야 겨우 원하는 결과를 얻어내고 있는 것이 현실인데, 이제부터 우리가 살펴볼 내용의 일부는 이와 같은 문제와 한계를 극복하려는 시도들이 어떤 식으로 이루어졌는지를 살펴보려는 것이기도 하다.

그렇다면, TEX에 남는 것은 무엇일까?

(1) **문서 조판 언어**라는 바로 그 특성이 TEX의 핵심이다. 하나의 언어로서 TEX은, 배우기가 쉽지 않고 조판이라는 한 분야에 지나치게 특화되어 있다는 치명적인 단점이 있기는 하지만 매우 견고한 매크로 언어이다. 따라서, 아무리 TEX의 확장이 있다 하더라도 그것은 TEX 설계의 근간인 "언어"로써 문서를 조판(typeset)한다는 그 개념을 벗어나지 않을 것이고, 그런 한 그것은 TEX일 것이다.

[3] *TUGboat*, vol. 23, 2002.

"문서를 프로그래밍한다는 것"이 주는 성취감은—비록 소수에게만 호소력이 있는 것일는지 모르나— 대단한 것으로서, TEX의 생명력의 한 원천이 되고 있다.

(2) 비록 낡았다 하나, H&J 알고리듬의 중요성을 소홀히 할 수는 없다. 하이프네이션과 저스티피케이션은 조판의 가장 기본적인 요소를 충족하는 것으로, 이것이 해결이 되어야만 그 위에 각종 기능을 추가해갈 수 있는 것인데, TEX은 바로 이것을 제공한다. 20년이 지나도록 아직도 상업용 조판 프로그램과 "비교될 수 있는" 알고리듬을 제시하고 있다는 점을 생각하면 이 점에서 TEX의 탁월성은 오히려 돋보이는 바 있다.

(3) 개방성. TEX을 확장하거나 변경하는 것이 용이했다 할 수 있는 이유는, 이것이 개방적인 공개 프로그램이기 때문이었다. TEX의 개방성은 누구든지 의사와 능력이 있다면 매크로를 추가하든 소스를 개선하든 하는 것이 열려 있다는 뜻이므로, 상업용 프로그램의 사적 소유물인 코드에 비교할 바가 아닌 것이다.

(4) 수식 조판의 표준. TEX이 오늘날까지 사용되고 있는 주요 요인 가운데 하나이다. 수식 표현의 문법은 TEX 문법을 바탕으로 하지 않을 수 없을 것이고 이 분야에서 TEX의 성능과 장점이 극대화됨을 부인할 수 없다.

3 TEX의 확장

현재 대부분의 TEX 배포판[4]이 채택하고 있는 "TEX이라 불리는 것(TEX 엔진)"은 실은 encTEX 패치된 pdf-ε-TEX이다. 이것은 다음 세 가지 TEX 확장을 결합한 것이라 할 수 있다.

pdfTEX + ε-TEX + encTEX

이 엔진의 채택은 teTEX 2.x 버전에서 3.x로 이행하는 과정에서 거의 표준화되었으며, 시기적으로 보면 2004년에서 2005년 사이가 이에 해당한다. 즉, 비교적 최근의 일이다. 우리가 명령행에서 `tex`이나 `latex`을 부르면 실제로는 pdf-ε-TEX이 TEX 호환성 모드로 실행되어 원하는 문서의 출력을 얻게 되는 것이다.

[4] Web2C에 기반한 배포판을 말한다. 예를 들면 teTEX, TEXLive, KC2006의 W32TEX, (그리고 부분적이지만) MiKTEX 등. 이 이외에도 PCTEX이나 VTEX과 같은 다른 TEX 배포판이 많다. 배포판에 관한 문제는 이 글에서 다룰 만한 것이 아니므로 다른 자리로 미뤄두기로 한다.

3.1 ε-TEX

ε-TEX이란, "extended(또는, expanded) TEX"을 가리키는 것으로, 유럽의 NTS 그룹에 의하여 만들어졌다. 현재 버전은 2004년에 발표된 v2.1이다.

ε-TEX을 만든 NTS 그룹에서는 TEX에 기초한 새로운 조판 시스템(New Typesetting System)을 만들어 내고자 하였다. 이를 위해 우선 Knuth-Standard TEX의 몇 가지 한계를 보충하는 작업을 시작하였는데, 그 결과가 현재 우리가 알고 있는 ε-TEX이라 할 수 있다. ε-TEX은 원래 의도였던 TEX 설계 자체를 리뉴얼하는 데까지 이르지는 못하였으나, Knuth가 (의도적이었을지?) TEX에 설정해두었던 여러 가지 제한을 실질적으로 제거하기 위한 상당히 많은 primitives들을 추가하였다.[5] 그 가운데는 오늘날의 관점에서 보았을 때 없어서 안 될 것들도 포함되어 있는 데다가 ε-TEX의 "normal" 모드는 실제 TEX과 똑같이 동작하기 때문에 이제는 TEX이라 하면 대부분 ε-TEX을 가리키는 말이 되었다고 보아도 좋을 정도이다.

그 가운데, 우리 입장에서 주목을 끄는 몇 가지만 두서없이 소개한다.

1. 양방향 조판(`TeX-XeT` primitives) Knuth 자신이 이미 선구적으로 기여한 바 있었던 `TeX-XeT`을 더욱 발전시켜, 역방향 조판을 가능하게 하였다.

2. register의 개수 제한 확대.

3. 디버깅을 위한 추가 매크로. 예컨대 assign, group, if, scantoken 등에 대하여 별도로 `\tracing`을 걸 수 있다.

4. 간결한 매크로 탐색. 예컨대 `\unless\expandafter\ifx\expandafter\relax\csname`과 같은 코드에서 발생할 수 있는 `\relax`의 side effect를 없애주는 새로운 매크로 검사 primitive `\ifcsname`을 쓸 수 있다.

이외에도 상당한 개선과 확장이 이루어졌으므로, 자세한 참고를 원한다면 ε-TEX Manual을 보라. 사용자 입장에서 이런 primitive를 직접 조작할 일은 많지 않기 때문에 실감하지 못하지만 사실상 "현재의 TEX은 TEX이 아니다"는 발언이 나오게 된 배경은 ε-TEX에 기원이 있다 해도 될 것이다.

[5] primitive란, TEX 설계의 근간이 되는 기본 매크로를 의미한다.

3.2 pdfTEX

ε-TEX이 여러 연구자들의 공동 작품인 데 비해, pdfTEX은 Hàn Thế Thành 씨의 연구 결과물이고 작품이다. 이것은 `.tex` 소스로부터 `dvi`를 거치지 않고 직접 `pdf`로 출력하도록 함으로써, `pdf`로는 구현할 수 있지만 `dvi`를 거치기에는 까다로운 복잡한 고급 조판 기능을 TEX에 부여하려는 목적으로 만들어졌다. 여기에 더하여 필요하다면 `.dvi`를 만들어내는 모드로도 동작하기 때문에 pdfTEX 엔진이 TEX 엔진을 대용할 수 있게 된 것이다. 처음에는 Web2C의 TEX 소스로부터 출발했지만 곧 ε-TEX을 받아들여, 현재의 pdfTEX은 사실상 pdf-ε-TEX으로 동작한다. 현재 pdfTEX은 Hàn 씨 외에 Martin Schröder, Hans Hagen 등이 참여하여 활발한 유지와 개발 활동을 하고 있다.[6]

pdfTEX은 아마 TEX에 프로그래밍 언어를 도입한 첫번째 사례로 기억되어야 할 것이다. TEX 자체를 폐쇄적인 언어로 보지 않고 TEX에 더 필요한 것을 오리지널 TEX보다 더 잘 할 수 있는 외부 언어를 통하여 구현함으로써 결과적으로 TEX을 확장할 수 있다는 개념은 이로부터 보편적으로 받아들여지게 될 소지가 생겨났다.

pdfTEX에서 주목해서 보아야 할 첫번째 요소는 역시 `pdf` 포맷 자체의 특성이다. pdfTEX은 pdf 언어를 거의 완전하게 제어할 수 있기 때문에, 그 표현 능력의 한계는 곧 `pdf` 언어 명세의 한계와 비슷하다. `pdf` 언어를 `tex` 소스로 제어할 수 있다는 것이 pdfTEX의 최대의 매력이라고 할 만하다.[7]

pdfTEX이 가진 또하나의 장점은 고급 조판 능력에서 기인한다. 위대한 디자이너라고 알려진 Hermann Zapf에게서 시작된 font expansion, margin kerning 등의 이른바 *micro-typography*를 TEX에서 구현할 수 있게 된 것은 특기해야 할 일이다.[8]

pdfTEX이 기여한 또하나의 중요한 측면은 TEX에 있어서 폰트 사용 측면의 확장이다. pdfTEX이 PostScript type 1 폰트를 다루는 방법은 놀라워서, 사실상 TEX

[6] 우리는 트루타입을 주로 사용하기 때문에 pdfTEX의 직접 혜택을 그다지 받지 못했다. 아직도 pdfTEX의 트루타입 지원은 몇 가지 측면에서 만족할 만하지 못하고 오픈타입 폰트에 대한 지원도 제한적이다. 이 문제는 pdfTEX의 다음 버전인 luaTEX에서 개선되리라고 기대하고 있다.

[7] 물론 DVIPDFM*x*라는, 조진환과 히라타의 `dvi` 드라이버도 `pdf` special 코드를 상당히 잘 처리한다. 이에 대해서는 조진환의 2005년 TUG Conference 발표 자료를 참고하라. 그러나 그 implementation은 `dvi` 드라이버 수준에서 이루어진 것으로, 우리가 문제삼고 있는 TEX 확장과 직접 관계가 적어서 이 글에서는 각주로 언급하는 데 그친다.

[8] 이 개념들을 한글 폰트에 얼마나 적용할 수 있는지는 모르겠다. 그러나 적어도 문장부호 등에 대해서는 이런 개념을 적용한다면 유용한 결과를 얻을 수 있을 터인데, 우연인지 모르겠으나 한중일 삼국 중에서 문장부호에 영문 폰트를 그대로 쓰는 나라는 우리나라밖에 없는지라, 용케도 마이크로타이포그래피의 효과를 우리는 시각적으로 체험할 수 있게 되어 있다. 이 문서에 pdfTEX의 margin kerning을 적용하였다. 사용한 폰트는 lmodern이다.

에 있어서 폰트의 제약을 —구미권에 한정된 일이지만— 상당한 정도로 극복하게 하였다는 점은 인정해야 할 것이다. 구미 언어권에서는 현재 `dvi`를 매개로 하는 TEX 작업 방식에서 pdfTEX이 직접 `pdf`를 generate하게 하는 방식으로 급격히 변해가고 있는 중이라고 한다.[9]

3.3 encTEX

encTEX은 Petr Olšák(체코)이 제작한 TEX 확장으로서 입·출력의 실시간 리인코딩(re-encoding)을 구현한 것이다. 특히 중요한 것은 UTF-8 인코딩을 포함한 multibyte 인코딩을 TEX이 직접 다룰 수 있도록 하였다는 점일 것이다. 현재 모든 유니코드 코드 포인트에 이르기까지 conversion table이 완성되어 있지는 않지만 앞으로 유니코드를 좀더 잘 다룰 수 있는 가능성이 생겼다는 데서 주목받고 있는 확장이라고 할 수 있을 것이다.

4 TEX 따라잡기

TEX의 '확장'과는 별개로 TEX을 다시 작성(re-write)하려는 시도들도 간간이 있었다. 그 가운데 두어 가지를 적어둔다. 이와 같이 TEX을 다시 쓰려는 시도가 직면하는 어려움은, 우선 TEX 수준의 동작이 에러 없이 이루어질 정도까지는 TEX을 앞서갔다고 하기 어렵다는 데 있다. 시도는 시도일 뿐이어서, 아직까지 획기적인 성과를 내었다고 할 만한 시스템은 보이지 않지만, 누가 알겠는가, 어느 날엔가 TEX을 능가하는 TEX-아닌 시스템에서 문서를 작성하게 되는 날이 올는지.

4.1 NTS: TEX Written in Java

ε-TEX을 만든 NTS 그룹에서는 이와는 별도로 NTS라는 이름의 Java Typesetting System을 공개한 바가 있다. 아직 β-버전인 이유는 TEX과 호환될 정도까지 개발이 진행되지 않아서인데, 최근 관심이 줄어든 듯하나, 여기 적어둔다.

[9] 최근 pdfTEX은 `cmap`을 직접 generate하는 새로운 기능을 실험적으로 도입하였다. 이 과정에서 우리나라에서는 조진환, 김도현 등의 기여가 있었는데, 이 문제가 우리의 TEX 사용 환경과 직접 관계가 있었기 때문이다. 이런저런 과정에서 pdfTEX에 새로운 기능도 추가되고 변화도 생겨나서, 한글 문서를 pdfTEX으로 작성하는 것이 그다지 어려운 일이 아니게 되었다고 할 수 있다. 이 자체가 최근 몇 개월 동안의 변화이다. 그 결과, 2005년에 `lshort-kr`을 번역하면서 pdfTEX을 폄하했던 내용이 사실이 아니게 되었다. 아마도 이런 것을 발전이라 할 것이다.

4.2 ANT

Achim Blumensath의 ANT[10] Project는, NTS와는 달리 TEX 구현을 따라잡거나 다른 언어로 흉내내려는 것이 아니고, TEX을 대체하는 새로운 typesetting mark-up 언어를 만들어내려는 것이다. 이 분이 TEX에 대해서 하는 말이 재미있다.

> TEX이 훌륭하지만, 배우기 어렵고 성가시다. 게다가 수학식 조판이라는 협애한 분야를 벗어나면 조판의 일반적 요구에 대응하기가 불가능하다. 한 예를 들자면 현재 LATEX의 새로운 출력 루틴은 무려 100페이지가 넘는다. 게다가 TEX의 성능을 확대하기도 매우 어려운 것이, 소스 코드가 도저히 읽을 수 없는 잡동사니이기 때문이다. 그래서 나는 심플하고 클린하고 modular한 디자인의 ANT를 새로 작성하기로 결심했다.

ANT가 지향하고 있는 것은, catcode 없는 쉬운 코드, 고수준의 스크립트 언어, 유니코드 지원, type 1과 트루타입 및 오픈타입 지원, 오픈타입의 고급 기능 지원, 칼라와 그래픽의 지원, 페이지 레이아웃 설정의 단순화 등이라고 한다. 이 자체만 놓고 보면 가히 TEX의 미래라고 할 만한데, 현재 개발 중인 프로그램이므로 그 귀추를 주목해 볼 만하다 하겠다.[11] 이 시스템은 주로 Ocaml이라는 언어로 개발되고 있다고 한다.

ANT의 일부 실험적 코드는 Omega와 결합하여 ANTOmega라는 매크로 패키지를 만들어내기도 하였다.

5 Omega를 둘러싼 이야기

한때는 `UTF-8`로 한글 문서를 작성하면 당시 `lambda`라는 Omega-LATEX을 실행해야 했다. TEX을 둘러싼 환경이 얼마나 급격하게 변하는지, 현재 Hangul-ucs를 중심으로 문서 작성을 하고 있는 우리 입장에서는 "굳이 그럴 필요가 있겠는가" 생각하지만, 아직도 위와 같은 질문과 답변이 이루어지는 곳이 없지 않다. 물론 HLATEX을 쓰지 않으면 안되는 상황—예컨대 HLATEX 자체를 테스트한다든가—도 있겠지.

HLATEX의 가이드였던 〈한글LATEX 길잡이〉라는 은광희 님이 작성한 문서에는 Omega에 대한 기대가 몇 페이지에 걸쳐 자세히 개진되어 있다. 즉, Omega는 TEX의 미래이며, 앞으로 TEX 시스템은 Omega로 발전하지 않을 수 없을 거라는 것이다.

---

[10] 유명한 말장난을 흉내낸 이름으로서, ANT is not TEX이라는 뜻이라고 한다.

[11] 현재 ANT로 조판된 결과물은, pdfTEX의 수준에 이르려면 해결해야 할 문제가 만만치 않아 보였다.

실제로 HLaTeX은 사실상 Lambda 시스템으로 발전했지만…, 바로 그 '발전' 때문에 쓸모가 없어졌다. 이 얘긴 좀 슬픈 이야기다.

5.1 Omega의 시작과 끝

Omega(Ω)는 생각건대 야심적인 TeX 리뉴얼 프로젝트였다. 프리미티브를 몇 개 추가하는 데 그친 것이 아니라, TeX 자신에게 multibyte 이해능력을 부여하려는 계획이었던 것이다. 20년전과 컴퓨팅 환경이 변했고 유니코드가 일반화된 지금, 이것은 너무나 합당하고 올바른 방향인 것처럼 보였다.

Omega는 1990년대 중반, John Plaice와 Yannis Haralambous에 의하여 개발이 시작되었다. 이 시스템이 가진 특징을 몇 가지로 요약하면 다음과 같다.

1. 문자와 포인터의 데이터 구조를 8비트에서 16비트로 확장하였다.
2. 프로그래밍 가능한 필터(otp)를 통하여 입출력을 제어할 수 있도록 하였다.
3. TeX Unicode 인코딩을 정의하였다(UT1, UT2).

Omega는 특히 아랍어, 히브리어 등 RL 언어권과 CJK 언어를 식자하거나 다국어 문서를 조판할 수 있는 기능을 갖추게 되어 그 발전 가능성에 기대가 컸다. `odvips`나 `oxdvi` 등 Omega가 만들어낸 `dvi`를 처리할 수 있는 기술도 발전하였으며, 만족스럽지는 않았으나 UT1 및 UT2 인코딩의 폰트도 일부 사용이 가능하게 되어 갔다. 특히 2003년에는 DVIPDFMx가 Omega의 `dvi`를 처리할 수 있는 유력한 드라이버가 되어 `pdf` 제작에도 방법이 생겨나게 되었다.

2003년과 2004년 사이에, Omega 프로젝트는 개발이 중단되기에 이른다. 개발 중단을 언급("발표"가 아니고)하던 자리에 마침 조진환 박사가 참석하고 있었는데, 청중들은 모두 "황당해 했다"는 전언이다. 이 석연치 않은 중단은 Omega에 많은 기대를 걸고 있던 아랍 및 CJK 언어권의 Omega/Lambda 사용자들을 실망하게 만들었다. 실제로 그 이후로 지금까지 Omega 시스템은 더이상 새로운 릴리스를 발표하지도 않았고, 차츰 논의의 중심에서 멀어져 가고 있는 것이 현실이다.[12]

Omega가 원래 의도하였던 완전한 TeX 다국어 환경으로의 이행은 충분히 실현되지 않았다. 그러나 `otp`라는 형태로 프로그래밍 가능한 TeX이 출현했던 것은, TeX

[12] 이것은 조금 다른 문제이기는 하나, 확실히 Omega는 "16비트 유니코드 TeX 시스템"이었다. 그러나 유니코드는 16비트가 아니다. 이 차이는 별것아닌 듯해도 엄청난 것으로서, 사실 16비트 문자밖에 다룰 수 없다면 진정한 의미에서 유니코드 시스템이라 말하기에는 한계가 없지 않았다 할 것이다. Omega의 유연성은 어떤 식으로든 이런 문제를 해결할 길을 열어둔 것이 사실이었으나 그 구체적 물증을 볼 수 없음이 안타깝다.

이 가진 한계를 극복하는 유력한 방법 가운데 하나로 보였다. 앞으로 어떤 형식이 되든 이 "프로그래밍 가능성"이 TEX의 발전 방향이 될 것임은 의문의 여지가 없다.

5.2 ε-Omega, or Aleph ℵ

Omega에 ε-TEX을 머지한 것을 ε-Omega라고 부른다. 나중에 Aleph로 명명되었다. Omega의 제작자들이 처음 의도하였던 프로젝트가 너무나 멀고 원대하여 단시일내에 사용자의 요구에 부합하기 어려워 보였기 때문에, "당장 사용가능한 Omega"를 ε-TEX과 결부시켜 제공한다는 계획으로 시작된 것이었다고 한다. Omega의 미래가 불투명한 지금, 아마도 Omega 프로젝트가 낳은 많은 자원을 실질적으로 사용할 수 있는 것은 Aleph를 통해서일 것이다. 다행히 Aleph는 느리지만 개선이 이루어져 가고 있고, 특히 ConTEXt가 Aleph 포맷을 지원하면서부터 그 사용가능성이 없지 않은 것으로 보아도 될 것이다.[13] 대부분의 TEX 배포판에서 `omega`를 실행하면 `aleph`가, 그리고 `lambda`를 부르면 `lamed`가 실행되게 되어 있다. 오리지널 Omega는 오늘날 거의 사용되지 않는다.

Aleph에 한글 문서 조판을 결부시키려는 시도는 아직 본격적으로 이루어지지 않았다. 비록 일부 bug-fix가 이루어졌다고는 하나, Omega 자체가 가진 불완전함이 적지 않았기 때문에, 아마도 이 시스템의 한글화 시도는 동시에 Aleph 자체의 개선과 거의 동의어가 될 가능성도 없다고 할 수 없다.

6 LATEX 3은 오는가?

LATEX은 TEX을 대표하는 매크로 패키지이다. LATEX을 그냥 TEX이라고 부르는 경우도 많다. 이제는 과학 문헌 작성의 표준 포맷으로 받아들여지고 있다. LATEX은 TEX을 조판 엔진으로 하는 "문서 작성 시스템"이다.[14] 즉, 일종의 워드 프로세서인 것이다. TEX이 프로그래밍 언어인 것과 대조된다고 하겠다. TEX 언어 그 자체로 조판을 못할 것은 없지만, 최소한의 매크로 패키지가 마련되어 있지 않으면 불편하기 짝이 없을 것이다. 그것은 마치 라이브러리 전혀 없이 raw C 언어만으로 복잡한 사용자 인터페이스와 내부 동작을 갖춘 CAD 프로그램을 만들어내라는 것이나 같

[13] Mem이라는, 다국어 LATEX 문서를 Aleph로 조판하는 패키지가 개발되고 있다. http://mem-latex.sf.net.

[14] 그러므로, 흔히 이루어지는 TEX과 MS Word의 비교는 잘못된 것이다. 굳이 비교를 하려 든다면 LATEX과 다른 워드(도큐먼트) 프로세싱 프로그램을 비교해야 할 것이다.

다. Application 작성자에게는 라이브러리가 필요하고, 문서 작성자에게는 매크로 패키지가 필요하다.

LaTeX을 만든 사람은 Leslie Lamport이지만, 현재의 LaTeX은 그의 손을 떠난 지 꽤 되었다. Lamport 자신도 TeX 관련된 분야를 떠났다.

6.1 LaTeX의 현 단계

우리는 Godot를 기다리는 사람들이다. 아무리 기다려도 LaTeX 3은 오지 않는다. LaTeX의 현재 버전은 LaTeX 2_ε인데,[15] 이 버전이 나온 지 10년이 지났고, 아마도 이 상태는 상당 기간 더 유지될 것으로 보인다. LaTeX 3 Project Team[16]이 하는 일은 LaTeX 2_ε를 유지하는 것이다. 일부 LaTeX 3 코드가 공개되어 있기는 하나, LaTeX 2_ε의 영향력이 너무나 크기 때문에 설령 오늘 당장 LaTeX 3이 나온다고 해도 과연 단기간에 LaTeX 3으로의 이행이 이루어질지 확신할 수 없다.

그러나, 확실한 것은 LaTeX 3 프로젝트 문서에서 이미 말하고 있는 바와 같이 LaTeX 2_ε가 가진 한계가 비교적 명확하다는 것이다. 기술적으로 pdfTeX과 같은 새로운 TeX이 나오기 전에 이루어진 LaTeX 2_ε에 새로운 기능을 추가하는 과정은 아무래도 코드를 복잡하게 만들고 어려운 사용자 인터페이스를 통한 우회적 방법에 의하지 않을 수 없었을 터이다. LaTeX 3은 이런 문제들을 근본적으로 해결할 방법을 찾고 있는 듯하다.

LaTeX의 가장 큰 강점은, 설계의 효율성도 아니며 매크로의 강력함도 아니요, 탁월한 기능이 있어서도 아니다. "많은 사람들이 사용한다는 것" 그것이 LaTeX을 LaTeX이게 하는 것이며, 그러다보니 수많은 솔루션이 집적되어 있다는 그 사실에서 "LaTeX은 편하다"는 말이 나오는 것이다. 게다가 결정적으로 각종 저널에서 LaTeX으로 기고를 받는다는 것이 LaTeX의 독점적 지배력의 원천이 되고 있다. LaTeX은 TeX 세계의 윈도우즈이다.

사실 우리가 실제 문서를 작성하는 과정에서 필요한 거의 모든 기능이 LaTeX으로 구현되어 있기는 하다. 그러나 그것은 LaTeX을 기반으로 한 추가 패키지를 통해서인 경우가 더 많으며, 첨단 기능일수록 더욱 그러한데, LaTeX의 설계 자체를 바꾸게 되면 이런 추가 패키지를 이용한 여러 해결책이 더 명료해지고 단순해질 수 있을 것이라는 기대를 가지고 있다. 아무튼 LaTeX의 수명은 길~ 것이 틀림없고, 그런 만큼 더 강력한

[15] 2005년 12월에 최신 버전이 업데이트되었다.

[16] Johannes Braams, David Carlisle, Robin Fairbairn

매크로가 되기를 바라 마지 않는다.

LaTeX 자체의 개발이 정체되어 있는 느낌을 주지만 응용 패키지의 개발은 매우 활발하다. 지금도 CTAN[17]의 `macros/latex/contrib` 아래는 가장 빠르게 갱신되는 영역이다.

6.2 한글LaTeX에 관한 짧은 노트

TeX 자체의 한글화는 시도된 바가 없지 않을 것이나 범용성을 획득하지 못했다. 가까운 일본의 경우는 이와 달리 pTeX이라는 "일본어 TeX"을 성공적으로 개발하여 정착시킨 것과 대조적이다. 아마도 일본어 記寫 체계가 우리와 달리 로마문자를 거의 사용하지 않았기 때문에 가능하지 않았을까 하는 짐작을 하는데, 유례가 적은 독특한 경우임은 틀림없다. 그 대가로, TeX 관련 시스템의 눈부신 발전에 유기적으로 적응하는 데 한계가 있을 테지만. 아무튼 CJK로 흔히 묶어서 하나의 문자체계권 비슷하게 처리하는 경우가 많은데, 우리나라의 글쓰기 방식은 CJ와는 다른 특성이 있고, 이것이 여러 가지 어려움을 가져온다. 예컨대 띄어쓰기라는 독특한 체계나 영문자를 문장부호로 차용하는 관행, 그리고 부족하고 부실한 자유 글꼴 등, 넘어야 할 벽이 장난 아니었다.

1990년대부터 시작된 LaTeX 매크로에 한글을 입히는 노력은 거의 결실을 거두었다. 현재 LaTeX 문서 작성에 있어 사용자 수준에서 어려움을 거의 느끼지 않는다. 우리 현실에서 한글 사용에 대한 노력이 LaTeX 매크로에 집중되어 있었던 것은 불가피했던 측면이 있다 하겠다. 그리고, 그 모든 노력은 대략 한글을 유니코드로 처리하는 방법으로 최종 결론이 내려진 상태이다.

여담이지만, 유니코드 텍스트를 처리하는 데 있어 LaTeX 자체는 아무 것도 해주지 않는다. LaTeX 패키지 가운데 하나인 `ucs` 패키지가 이 문제에 대해 잠정적 해결책을 제시하고 있으나, 만약 유니코드 자체를 지원하는 TeX 시스템이 범용으로 사용되게 된다면 `ucs`보다 나은 해결책을 활용하게 될 수도 있을 것이다. LaTeX-ucs가 편리하고 필요에 부응하는 것이기는 하나, 그 architecture에 있어 상대적으로 아름답다고 하기 주저되는 것은 사실이다.

[17] Comprehensive TeX Archive Network. http://ctan.tug.org/tex-archive/

6.3 ConTEXt라는 것

LaTeX의 한계를 새로운 매크로 패키지로 넘어서려 한 시도들은 기존에도 많았다. HTML 문법과 비슷한 mark-up 방식을 채용한 StarTeX과 같은 특이한 것도 있었고 그밖에도 BLUeTeX과 같은 독자적인 매크로가 명멸했지만, LaTeX에 필적할 만한 사용자군을 형성하는 데는 거의 모두 실패했다.

유일하게, 사용자군이 넓어져가고 있는 새로운 매크로 패키지가 바로 Hans Hagen 씨의 ConTeXt이다. 다른 매크로와 달리 이 패키지는 처음부터 LaTeX을 전혀 의식하지 않고 완전히 새로운 매크로 코딩 방식을 채용하여 발전하였다.

ConTeXt의 새로움은 여러 측면이 있지만, 특기할 만한 것은 처음부터 `pdf` 출력만을 타겟으로 제작되었다는 것이다. `dvi`를 만들지 못할 것은 없으나, `pdf`를 제작하기 위한 중간 단계로만 활용하기 때문에 종래의 `dvi`와는 그 용도가 같지 않다.[18] 따라서 ConTeXt는 설계부터 pdfTeX에 크게 의존했고 pdfTeX의 발전이 ConTeXt에 의해 추동된 측면도 크다. 또한 pdfTeX의 발전으로 ConTeXt 자체도 크게 변모할 수 있었다. 현재 ConTeXt 제작자인 Hans Hagen 씨 자신이 pdfTeX 팀의 일원이 되어 있음을 생각하면 이것은 어쩌면 당연한 결과인지도 모르겠다.

아마도 TeX의 설계상 엄격함을 매크로 패키지 수준에서 넘어서려 시도한 것은 또다른 특징이라 할 수 있을 것이다. TeX의 pagebreak 알고리듬은 상당히 rigid 해서 한번 shipout한 페이지에 대해서 재검토하지 않는다. ConTeXt는 TeX을 반복 호출하는 불편을 감수하면서라도 이런 rigid한 알고리듬을 더 유연하게 바꾸기 위한 노력을 감행했다. 이런 면면은 pagebreak 뿐 아니라 개행 알고리듬에 이르기까지 다양한 방면에 걸쳐 있다 한다.

또한, ConTeXt는 조판 상의 요구를 더 효과적으로 mark-up 언어로 구현하는 데 관심을 가졌다. ConTeXt의 많은 화려한 예제들은 종래 레이아웃 프로그램으로나 구현할 수 있을 것으로 여겨졌던 효과들을 TeX 코딩으로도 할 수 있음을 보여준 놀라운 것이 많다. 또한, MathML이나 METAPOST, ruby 및 lua와 같은 외부 언어를 깊숙히 차용하여 본 경험은 아마도 다음 세대 TeX의 발전에 귀중한 자산이 될 것이다.

일반 사용자 입장에서 ConTeXt는 아직은 안정화된 매크로로 보이지 않는 면이

[18] 유구한 역사와 전통의 `dvips`는 이제 그 수명을 다한 것인가? 현재 `dvips`는 pstricks라는 강력한 그래픽 툴을 위해서만 쓸모가 있다고 보는 것이 틀리지 않는다. 아마 pstricks 패키지도 머지 않은 장래에 그 대응물에 의하여 대체될 가능성이 없지 않고 보면 TeX의 PostScript 출력 기반은 와해되어 간다고 해도 과언이 아닐 것이다. 이것은 필자의 개인적인 의견이고 전망이기 때문에 각주에 적어둔다. 아무튼 300dpi 프린터용 비트맵 폰트를 만들어서 찍던 시대가 아닌 것은 확실한 듯하다.

있을 것이다. 그러나 ConTEXt garden[19]을 방문하여 보면, 이미 상당한 정도로 문서 작성에 큰 불편이 없을 만큼 발전해 있음을 알게 될는지도 모른다. 특히 화려한 스크린 pdf 예제들이 눈길을 끈다.

7 TEX과 폰트 문제

오리지널 TEX의 설계에서 재미있는 것 하나가, METAFONT라는 폰트 기술 언어를 함께 만들어낸 것이었다. TEX이 사자로 상징되고 METAFONT가 암사자로 상징되는 유명한 일러스트레이션에서도 그 긴밀한 관계를 알 수 있다. METAFONT는 TEX의 일부가 아니라 그 자체로 독립된 시스템이었으며, TEX이 폰트를 처리하는 기술이던 tfm을 만들고 비트맵으로 번역(컴파일) 해주는 유틸리티와 함께 제공되었다. METAFONT는 매우 재미있는 윤곽선 글꼴 기술 언어였지만 그 실제 사용에 있어서는 특정 프린터에 맞는 비트맵으로 변환해야지만 출력을 볼 수 있었다.

어찌되었든, TEX 자체는 tfm이라는 폰트 메트릭을 이용하여 조판한다. 그 이후의 많은 TEX 확장들도 이 TEX의 기본 설계를 재검토한 것은 없다. TEX-류에서 tfm 폰트 처리 기술은 당연한 것으로 받아들여지고 있었다. 이것은 TEX의 특징이고 장점이면서 동시에 한계로도 작용하였다. 이 한계는, TEX이 METAFONT를 벗어나서 PostScript 글꼴을 받아들이면서부터 문제가 되었다. PostScript에는 afm 또는 pfm이라는 별도의 메트릭이 있음에도 불구하고 tfm을 일일이 별도로 만들어주어야 했던 것이다. 많은 솔루션이 제공되어 거의 불편함을 느낄 수 없을 정도가 되기는 하였어도 역시 폰트 자체에 디자인된 정보를 이용하지 못하고 별도의 tfm을 매번 만들어 제공해야 하는 것은 번거로운 일이었다.[20] 폰트 부분에 있어 TEX 자체에 약간의 확장이 가해짐으로써 다양한 폰트 활용이 가능해지기는 했지만, 여전히 폰트 문제는 TEX의 가장 TEX다운 측면으로 남아 있다. TEX의 설계가 tfm에 워낙 밀착해 있기 때문이다. 그러나…, 오픈타입 폰트와 같은 새로운 폰트 설계를 tfm과 연계하는 것은 어렵기도 하거니와 폰트 자체에 프로그래밍된 meta-data를 포기해야 하는 문제점도 동시에 가지고 있는 것이다.

[19] http://wiki.contextgarden.net

[20] 이 사태는 특히 우리와 같은 CJK 폰트를 이용할 때 더욱 현저하다. 예컨대 하나의 한글 폰트를 서브폰트로 취급한다 할 때, 도대체 몇 개의 tfm을 만들어내어야 하는가.

7.1 XeTeX이란 무엇인가?

XeTeX은, SIL international 社의 Jonathan Kew에 의하여 제작된 새로운 개념의 typesetting system이다. 처음에는 매킨토시 Mac OS X에서만 실행되도록 개발되었다가, 최근 Linux와 Windows 포팅이 차례로 발표되었다.[21] 원래의 매킨토시 버전은 ATSUI라는 라이브러리를 써서 구현된 것이었다는데, Linux와 Windows 버전은 Freetype을 채용하고 있다.

이 시스템은 폰트와 입력을 처리하는 방법에서 TeX의 처음 디자인에의 의존성을 제거하였다. 즉, 모든 "문자"는 폰트에 대응하여 식자되며 그 식자규칙은 유니코드 표준을 따른다. `tfm`은 더이상 필요하지 않다. 오직 잘 디자인된 고품질의 폰트만 있으면 되는 것이다. 이 폰트에 대한 정보는 폰트 자체 또는 그 폰트를 해석하는 시스템과 라이브러리에 의존한다. 이것이 TeX 프레임워크 내에서 동작하는 한, `tfm` 중심의 기존 TeX 폰트에 호환성이 유지되겠지만, 그보다는 폰트 사용의 혁신이 훨씬 돋보이는 바 있다. 오픈타입 글꼴과 같은 신기술을 TeX에 도입하는 새로운 선례를 만든 것으로, 현재로서도 폰트에 관한 한, 이만큼 혁신적인 TeX 시스템은 없었다.[22]

그런데, 문제는 이것이 과연 TeX인가 하는 것이다. 폰트 사용 설계를 배제하고도 TeX이라고 부를 수 있다면 TeX이란 과연 무엇일까? XeTeX은 여전히 mark-up 방식의 소스를 처리하며, TeX 명령과 매크로에 의하여 동작한다. 기존의 LaTeX 코드들도 잘 해석해서 식자해준다. ConTeXt에서 XeTeX을 엔진으로 하는, XonTeXt라고 불리는 시스템도 상당히 원활하게 작동한다. 그러나 내부적으로 문자와 폰트를 처리하는 방식은 TeX에 의존하지 않고 있다.

7.2 XeTeX의 가능성

우리는 XeTeX의 가능성에 상당한 기대를 걸지 않을 수 없다. 이 시스템을 잘만 성장한다면, 그동안 한글 문서 작성에서 겪었던 많은 문제들이 단숨에 해결된다.

---

[21] Windows 포트는 W32TeX에 포함되어 있다. 당연히 KC2006에도 포함되어 있는데, MiKTeX에 집착하는 많은 Windows 사용자는 이 시스템을 아직까지는 구경해보지 못하고 있다. 필자는 MiKTeX이라는 시스템이 Windows TeX 사용자를 오도하는 측면이 있다고 생각하며, MiKTeX에의 종속에서 벗어나는 것이 시급한 일이라고 판단하고 있다. (이 판단은 당연히 언제든 바뀔 수 있는 것이다.)

[22] 물론, 모든 문제가 다 해결된 것은 아니다. 특히 한글 문서를 XeTeX으로 식자하는 데서 부딪치는 근본 문제는, 신뢰할 만한 고품위의 디자인을 갖춘 한글 폰트가 정말 드물다는 것이다. 예컨대 웬만한 성가를 얻고 있는 상업용 폰트의 경우에도 영문자 디자인이 성의없이 처리되거나 적절한 리가쳐와 커닝 정보를 포함하지 않음으로써, 참고 보기 힘든 판면을 만들어내게 되는 것은 시스템의 잘못은 아닌 것이다.

예컨대, 비록 트루타입을 사용할 수 있게 되었다고 하나 그 복잡했던 절차를 모두 생략할 수 있다는 것, 또한 앞으로 발전할 새로운 폰트 기술을 TEX 문서 작성에 통합할 수 있다는 것 등, 할 수 있는 일이 많다. 그러나 아직은 많은 부분이 미래를 위해 유보되어 있다 할 것이다.[23]

8 TEX의 미래 : programmable TEX

프로그래밍 언어와 TEX의 만남은 그 자체로 하나의 토론 주제가 될 것이다.[24] 이 글에서 이 문제를 근본적으로 다루는 것은 아마 불가능할 것이다.

여기에는 두 가지 방향이 있겠다.

(1) 매크로 수준에서 프로그래밍 언어를 활용하려는 것.

(2) TEX 자체를 확장하는 수단으로 TEX의 언어적 유연성을 높여 프로그래밍 가능하게 만드는 것.

PerlTEX이나 SQL-TEX이 첫번째 방향을 대표하고 있다. 이것은 TEX 시스템 자체를 개선하거나 확장하는 것이 아니라, 특정 언어로 구현된 내용을 마치 TEX 매크로인 것처럼 사용하게 해주려는 것이다. 이것도 매우 재미난 결과를 얻을 수 있고, 특히 사용자 인터페이스에 밀착한 것이니만큼 사용자 입장에서는 더 흥미로울 수 있다. PerlTEX을 이용하여 정렬이나 데이터 처리를 구현한 예가 KTUG에 소개되어 있기도 하다.

우리가 이 글에서 관심을 가지는 방향은 TEX 언어를 개선하고 처리 과정을 재편하는 것으로 두 번째 것에 해당한다. 최근 유력한 것은 lua라는 프로그래밍 언어를 TEX과 통합하는 이른바 luaTEX 프로젝트이다.[25]

왜 프로그래밍 가능한 TEX이 필요한가? TEX의 유연성을 위해서는 물론이고, 특히 오픈타입 폰트의 활용을 위해서 필요하다고 할 수 있다. 오픈타입이라는 폰트 스펙 자체가 프로그래밍 가능한 것이므로, 그것을 이용할 TEX도 당연히 프로그래밍이

[23] 그 전에 필자는 개인적으로, 이 모든 신기술을 테스트할 수 있는 고품질의 오픈타입 글꼴 한 벌을 보게 되는 것이 소원이다. 한글 TEX 발전을 가로막고 있는 여러 요인 중에는 상업적 폰트 제작회사와 출력소의 이해관계가 한몫을 하고 있다고 판단하지 않을 수 없다. 이것은 물로 가는 자동차를 만들어도 정유회사의 이해관계 때문에 판매 금지를 피할 수 없을 것이라는 어느 분의 말씀을 생각나게 하는 점이 있다.

[24] KTUG 컨퍼런스나 연구발표회가 열린다면 언젠가는 주제로 등장할 것이 틀림없다.

[25] 한때 ruby가 적극적으로 검토된 적이 있었다고 하나, 최종 결론은 lua 쪽으로 가는 것으로 난 모양이다.

가능해야 할 것이다. 현재 그런 종류의 오픈타입이 과연 있느냐는 별개의 문제라고 생각한다.

TEX에는 그래픽 처리 언어가 없다.[26] TEX 친화적인 그래픽 언어로 pstricks나 METAPOST가 있는데, 이런 언어를 내재적으로 활용하여 문서를 작성해본 경험이 있다면 TEX의 확장성에 감탄을 금할 수 없을 것이다. TEX 자체에 언어 처리가 가능하게 하는 것은 어떤 의미에서는 필수적인 면이 있다.

8.1 luaTEX

TEX과 프로그래밍 언어의 결합이 구체적으로 추진되고 있는 프로젝트가 있다. ConTEXt와 pdfTEX진영은 그 다음 차기 버전으로 luaTEX을 준비하고 있다고 한다. 이것은 lua라는 가벼우나 강력한 프로그래밍 언어와 TEX을 결합하여 TEX의 정체된 언어적 능력을 보강하려는 계획이다. 간략히 표현하면 lua-embedded TEX language의 탄생을 눈앞에 두고 있는 셈이다.

아마도 luaTEX은, 'TEX으로 할 수 있는 모든 것'과 'lua로 할 수 있는 모든 것'을 결합하게 될 것이다. 생각만 해도 즐거운 상상이지만, 문제라면, lua라는 스크립트 언어를 새로 배워야 한다는 거~.

알려진 바에 따르면 pdfluaTEX은 pdfTEX의 후신이 될 것이고, 미루어진 많은 문제, 예컨대 오픈타입에 대한 보다 근본적인 지원 등도 luaTEX 프로젝트에 포함될 것이라 한다.

왜 perl이나 ruby가 아니라 lua였을까? 이 질문에 대해서 luaTEX 사이트에서는, free–embeddable–small–portable–easy–fun이라고 답변하고 있다.[27]

현재 pdfTEX의 영향력이나 ConTEXt의 확장성과 발전가능성을 볼 때, luaTEX은 상당한 영향력을 가지게 될 가능성이 높아 보인다. 그러나 일반 사용자, 즉 말단 문서 작성자에게 lua 언어의 습득을 요구하는 일은 아마 없을 것이다. 설계하는 사람과 코딩하는 사람의 분리, 그리고 디자인하는 사람과 글쓰는 사람의 분리는 LATEX만이 아니라 TEX이라는 매크로 언어를 통한 문서 조판에서 면연히 흐르는 하나의 근본 합의라고 생각한다.

[26] LATEX의 `picture` 환경은 폰트 기반 환경이지 그래픽 언어가 아니다.

[27] http://www.luatex.org/faq.html

9 結 : 한글 사용자의 TEX 쓰기

원래 이 글의 초안에는 XML과 TEX, 그리고 TEX과 그래픽 문제를 다루려는 두 개 절이 더 있었다. 시간의 제약과 능력의 부족으로 이 문제를 언급하지 못하였는데, 훗날 다시 논의할 기회가 있기를 바란다.

지금까지 우리가 살펴본 것은, TEX 또는 그 주변 프로그램들의 현재 상황이 대단히 역동적이라는 것이다. 흔히 알려진 것과 달리 TEX은 정체된 프로그램도 아니요 해결해야 할 문제가 없는 프로그램도 아니다.

이러한 역동적 상황이 창출된 가장 큰 계기는 역시 유니코드의 등장이었다. 어떤 사람이 유니코드를 다루지 못하는 것을 "TEX의 설계상 하자"라고 하고 말하는 것을 들은 바 있지만, TEX이 설계되던 70년대 당시 유니코드를 고려할 이유도 방법도 없었던 것이다. 유니코드를 TEX이라는 조판 프로그래밍 언어와 조화시키려는 노력은 다양한 방식으로 전개되어 왔고, 우리는 그 과정을 훑어볼 수 있었다.

또 하나의 중요한 계기는 폰트 문제이다. 오리지널의 Knuth-standard TEX에서 상정한 폰트 시스템과는 비교할 수 없는 "폰트라는 새로운 세상"이 펼쳐진 것은 TEX의 설계가 종료된 이후였다. 현재 주요 이슈는 역시 오픈타입이고 이미 말한 대로 이 오픈타입을 적절하기 다루기 위한 준비가 진행되고 있다고 해도 될 것이다. 오픈타입을 취급하는 것이 한결 쉬워지면 아마도 한글 사용에 있어서 새로운 돌파구가 열릴 수도 있을는지 모른다는 기대를 하여 본다.

원래부터 다국어 설계가 아니었던 TEX으로, 한글 문서를 훌륭하게 조판해낼 수 있게 된 지금, 한편으로는 TEX 시스템 자체를 개선하려는 수많은 노력과, 다른 한편으로 한글에 특화된 여러 측면을 구현하려는 노력이 결합된 결과라는 결론을 잠정적으로 내릴 수 있겠다. 한글이라는 특정 문자의 식자와 조판에 있어서 TEX은 이제야말로 그 가능성의 일면을 내포하게 되었다고 할 수 있겠으며, 그런 의미에서 우리는 새로운 출발선에 서 있다고 생각한다. 해결해야 할 숱한 문제가 우리를 기다리고 있는 것이다.[28]

Knuth 박사의 *The Art of Computer Programming*의 한국어 번역본이 출간된다는 반가운 소식과 더불어, TEX의 출판 적용 가능성에 대한 회의적 시각으로 인해 한글(HWP)로 이 책을 번역하게 되었다는 슬픈 소식이 전해지자, KTUG 내부적으로 약간 의기소침해진 일이 있었다. 이 책조차 한글 TEX으로 처리하지 못한다면,

[28] 그러나 동시에, 현재 요구되는 한글 문서의 타입세팅에 TEX이 충분히 대응할 수 있을 정도가 되어 있다는 것을 자랑스럽게 생각한다.

지금까지 해온 모든 것이 다 쓸데없는 일이 아니었을까?

어쩌면 아직도 TEX으로는 한 가지 서체밖에 쓸 수 없다든가, “똠방각하”를 찍을 수 없다든가, 확장한자를 사용할 수 없다든가, 자간 행간 조절이 안된다든가, 글자 모양이 출판물에는 적합하지 않다든가 하는 수많은 “상식화된 오해”를 극복할 만한 결과물을 납득할 만하게 제공하지 못한 잘못도 없지 않은 것일 터이다. 아니면, TEX이 아직도 개선되고 발전하고 있는 새로운 프로그램임을 알리지 못한 것이거나.