LaTeX의 즐거움

첫 번째 이야기

nova de hi

2024년 5월 20일

세종과학예술영재학교 모.텍.동. 세미나

들어가기

용어

TEX	‘텍’이라고 읽는다. ‘텍스’라고 읽지 않는다.
LATEX	‘라텍’ 또는 ‘레이텍’이라고 읽는다. ‘라텍스’가 아니다.
TEX Live	텍을 사용할 수 있게 해주는 프로그램 집합. *텍 시스템* 또는 *텍 구현*.
MiKTEX	TEX Live와 같은 텍 프로그램 집합. 특히 Windows 친화적이다.

용어 (계속)

컴파일 plain text로 입력된 문서 원본을 '처리'하여 출력물(=pdf)로 변환하는 과정을 비유적으로 일컫는 말. 소스를 컴파일하는 프로그램을 컴파일러라고 한다.

소스 문서의 원본. 컴파일러에게 전달하는 입력 파일이 된다. 플레인 텍스트로 되어 있다. 한글이 포함되어 있다면 UTF-8 인코딩을 적용한다.

엔진 텍 근원 프로그램. 나중에 좀더 자세히 살펴보자.

에디터 원본 파일을 편집하는 응용 프로그램.

용어: LaTeX

클래스 LaTeX이라는 규격에서 반드시 요구하는 문서 기본 형식 정의 라이브러리.

패키지 LaTeX의 기능 확장을 위한 라이브러리.

명령(어) 텍의 콘트롤 시퀀스를 레이텍에서 명령이라고 한다. 인자 없는 명령을 특별히 '선언'이라고 한다.

환경 입력 문자열의 일부를 `\begin`으로 시작하여 `\end`로 끝나는 블럭으로 가두어서 일정한 효과를 적용하는 범위.

예약문자 텍이 입력문자열을 해석할 때 특별히 사전에 약속한 문자. \, #, %, $, _, ^, &, {, }, ~ 등이 있다.

엔진

- TEX은 Donald Knuth라는 컴퓨터학자 겸 수학자가 1980년대에 완성한 조판 프로그램(typesetting program)이다.
- 초기에는 오직 ASCII 문자만을 독자적으로 고안된 폰트로 식자할 수 있었다. (1982–3년경)
- TEX의 제한된 메모리 사용을 극복하고 많은 추가 기능을 붙여 확대한 확장 엔진 ε-TEX.
- 출력 형식을 pdf로 하고 미세 타이포그래피를 적용한 pdfTEX
- 문자 인코딩을 유니코드로 하고 트루타입 오픈타입 폰트를 적용한 XƎTEX
- Lua 언어를 매크로와 함수 정의에 직접 활용할 수 있게 한 LuaTEX

엔진

- TEX은 Donald Knuth라는 컴퓨터학자 겸 수학자가 1980년대에 완성한 조판 프로그램(typesetting program)이다.
- 초기에는 오직 ASCII 문자만을 독자적으로 고안된 폰트로 식자할 수 있었다. (1982–3년경)
- TEX의 제한된 메모리 사용을 극복하고 많은 추가 기능을 붙여 확대한 확장 엔진 ε-TEX.
- 출력 형식을 pdf로 하고 미세 타이포그래피를 적용한 pdfTEX
- 문자 인코딩을 유니코드로 하고 트루타입 오픈타입 폰트를 적용한 XƎTEX
- Lua 언어를 매크로와 함수 정의에 직접 활용할 수 있게 한 LuaTEX

요약

영어 문서 작성에는 pdfTEX이, 한글 문서 작성에는 XƎTEX 또는 LuaTEX이 적합하다.

철학

플레인 텍스트의 세계

선언

플레인 텍스트 문서 작성은 '기술'의 문제가 아니라 '철학'의 문제이다.

간이성 플레인 텍스트 파일은 쉽게 읽고 편집하고 배포할 수 있다.

이식성 어떤 컴퓨터에서도 원칙적으로 플레인 텍스트 파일을 읽을 수 있다.

보존성 이론상 백 년 후의 컴퓨터에서도 읽을 수 있다.

표준성 플레인 텍스트 파일의 문자집합과 그 인코딩 방식은 표준에 적합하다.

조작성 플레인 텍스트 파일을 대상으로 어떤 컴퓨터 언어로든 쉽게 다루고 프로그래밍할 수 있다.

문서 생성과 플레인 텍스트

플레인 텍스트에서 양식화된 문서를 만드는 방식을 *Markup* 문서 작성이라고 한다.

LaTeX	The Game itself.
MarkDown	간단한 문법으로 이루어진 간이 문서 작성 언어. *pandoc*이라는 유틸리티로 HTML, TeX, docx (MS Word) 등의 포맷으로 변환할 수 있다.
XML	플레인 텍스트 문서 작성의 사실상 표준. 풍부하고 세밀한 규정이 가능하나 복잡하고 확장성이 너무 높아서 보통 다른 유틸리티(=워드프로세서)를 경유하여 활용한다. Word의 `docx`, HWP의 `hwpx` 등도 결국은 XML 포맷이다.
DOCBOOK	XML에 기반한 문서 기술 마크업 언어

LaTeX으로 할 수 있는 것

구조화된 문서

구조화된 문서 또는 **문서 구조화**란 내부 논리적 흐름에 따라 적절한 *division* (장절편제)을 갖추고 이들 사이에 유기적으로 통합된 문서를 생성하는 것을 말한다.

- 장절편제는 *level*로 구분된다. `\chapter`, `\section`
- 장절로 문서를 *편제(describe)*하는 것과 그것을 *현시(display)*하는 것은 다른 문제이다.
- 주로 논리적 문서, 긴 문서(도서, 논문, 보고서), 내적 논리가 중요한 문서에서 채택된다.
- 이에 대하여 *즉흥적* 문서는 예를 들면 메모, 편지, 초대장 등 장절편제를 갖지 않는 문서를 말한다.

구조화된 문서

구조화된 문서 또는 **문서 구조화**란 내부 논리적 흐름에 따라 적절한 *division* (장절편제)을 갖추고 이들 사이에 유기적으로 통합된 문서를 생성하는 것을 말한다.

- 장절편제는 *level*로 구분된다. `\chapter`, `\section`
- 장절로 문서를 *편제(describe)*하는 것과 그것을 *현시(display)*하는 것은 다른 문제이다.
- 주로 논리적 문서, 긴 문서(도서, 논문, 보고서), 내적 논리가 중요한 문서에서 채택된다.
- 이에 대하여 *즉흥적* 문서는 예를 들면 메모, 편지, 초대장 등 장절편제를 갖지 않는 문서를 말한다.

요약

LaTeX은 본질적으로 문서를 구조화하여 작성하는 데 적합하다.

수학식

1. 수학식을 표현하는 데 TEX 문법, 특히 *amstex*은 사실상의 표준 (*de facto* standard)이다.
2. 다른 표준화 방안으로 *MathML* 같은 것이 꾸준히 논의되어 왔으나 결국 이를 벗어나지 못했다.
3. 문서에서만이 아니라, 웹페이지(HTML)에서 수식을 쓸 때, 이메일에서 수학적 논의를 하고자 할 때, 또는 수학식을 발표해야 할 때, LATEX 수식 마크업을 쓰지 않을 수가 없다.
4. 다른 것을 모두 포기하더라도 수학식 표현만큼은 반드시 배워두어야 한다.

수학식

1. 수학식을 표현하는 데 TEX 문법, 특히 *amstex*은 사실상의 표준 (*de facto* standard)이다.
2. 다른 표준화 방안으로 *MathML* 같은 것이 꾸준히 논의되어 왔으나 결국 이를 벗어나지 못했다.
3. 문서에서만이 아니라, 웹페이지(HTML)에서 수식을 쓸 때, 이메일에서 수학적 논의를 하고자 할 때, 또는 수학식을 발표해야 할 때, LATEX 수식 마크업을 쓰지 않을 수가 없다.
4. 다른 것을 모두 포기하더라도 수학식 표현만큼은 반드시 배워두어야 한다.

요약

수학식의 기술과 표현은 LATEX이 갖는 최고의 장점 중 하나이다.

문서의 미학

1. 수학식의 미학적 표현은 레이텍으로 조판한 문서가 **압도적으로** 지지를 받는데, 그 이유는 많은 수학 관련 학자 학생들이 이 수식 표현에 너무 익숙하기 때문이다.
2. 대체로 과학적 문서에서 LaTeX으로 작성한 문서가 **미학적으로** acceptable한 것으로 평가받는다. 특히 한국어 문서에서 이것은 활용 능력이 비약적으로 발전한 최근의 일이다.
3. 도서의 구성하는 다른 속성, 예컨대 도표/도안/그림 등을 매우 쉽고 논리적으로 다룰 수 있다.
4. 이미 존재하는 문서의 유지와 관리 및 재편이 어떤 다른 도구보다 조작가능성이 높다.

프로그래밍 가능성(Programmability)

프로그래밍 가능성이란 다음 두 가지 의미가 있다.

1. **외적 유연성** 텍스트를 처리하는 다른 프로그램이나 유틸리티, 데이터베이스 프로그램 등에서 최종 출력 문서를 생성하는 프로그래밍이 가능하다. 예를 들면 python의 pynotebook, R의 rmarkdown, SQL 데이터베이스의 출력물 제작 등을 해당 프로그램 내부에서 조작할 수 있다. 예컨대 SQL이 제공한 주소록 명단을 출력하려 할 때, 최종 출력 포맷이 HWP인 경우를 상상할 수 있겠는가?
2. **내적 자유도** 문서 생산만을 문제삼더라도, 예컨대, “다섯번째 단어마다 배경색을 무작위로 바꾸어라”와 같은 요구에 워드 프로세서로 대응할 방법이 있나? 상호참조, 자동조사, 문헌인용 등 LaTeX이 자랑하는 대부분의 기능은 모두 *프로그래밍*의 결과이다.

무엇을 얼마만큼 공부해야 할까?

For What?

- 학위논문 또는 졸업논문을 작성하기 위한 목적
- 단행본을 출판하기 위한 목적

For What?

- 학위논문 또는 졸업논문을 작성하기 위한 목적
- 단행본을 출판하기 위한 목적
- LaTeX 프로그래밍 자체를 즐기기 위한 목적
  - 스타일 패키지 제작이 가능할까?
  - 자신이 관리하는 홈페이지에서 TeX을 이용한 동적 문서생성 기능을 프로그래밍할 수 있을까?
  - 다른 학생들의 졸업논문 작성을 쉽게 만들어주는 방법이 없을까?

For What?

- 학위논문 또는 졸업논문을 작성하기 위한 목적
- 단행본을 출판하기 위한 목적
- LaTeX 프로그래밍 자체를 즐기기 위한 목적
  - 스타일 패키지 제작이 가능할까?
  - 자신이 관리하는 홈페이지에서 TeX을 이용한 동적 문서생성 기능을 프로그래밍할 수 있을까?
  - 다른 학생들의 졸업논문 작성을 쉽게 만들어주는 방법이 없을까?
- 뭔가 있어 보여서 자존감을 보충하기 위한 목적
  - 유튜브에 LaTeX 관련 콘텐츠를 제작하여 올리면 과연 구독자가 몇 명이나 될까?
  - 과연 “나 레이텍 좀 알아”라고 하면 후배들이 멋있다고 생각해줄까?

매뉴얼을 읽자

- 2024년 현재, 대부분의 상상가능한 기능들은 패키지라는 것으로 구현되어 있다.
- 문제는, 어떤 패키지가 내가 필요로 하는 것인지 알기 어렵다는 것.
- 패키지에 대한 정보를 얻었으면, *package documentation* 이라는 문서(대부분 pdf)를 읽어야 한다.
- *texdoc*이라는 프로그램을 활용하자.
  - *kstexdoc*은 *KTUG 사설저장소* 패키지에 대한 문서를 읽기 쉽게 하기 위한 것이다.

질문하고, 답변을 받자

- 패키지에 대한 정보를 어디서 얻을 것인가?
  - ChatGPT, copilot 같은 애들한테 질문해서 얻은 답은 (아직?) 완벽하지 않다. 예제라도 보여주는 게 거의 대부분 쓸모없거나 잘 안 돌거나……
  - 주변의 guru (=nerd)에게 질문하자
  - 검색을 생활화한다. google을 통해서, stackexchange를 통해서, ktug을 통해서……
- 최후의 수단은 ktug 질문 게시판. 샘플 예제 첨부만 잘 하면 대부분 원하는 답변을 얻을 수 있는 곳이다.

마지막 말

당부의 말: 재미?

1. 플레인 텍스트 문서 작성의 철학에 익숙해지는 것이 좋다. 굳이 LaTeX이 아니더라도 *markdown* 같은 것을 생활화해두면 언젠가는 반드시 도움이 된다.
2. 문서를 코딩하는 것은 꽤 재미있는 일이다. 어떤 부분의 코드를 바꾸면 어떤 결과가 나타날지 예측하고 의도대로 되었을 때 성취감을 느낄 수도 있다.
3. 남들이 잘 아는 것만 배우다가는 평생 남들보다 더 알게 되는 것이 없다. 좀 희귀한 것을 배워두면 혹시 자랑할 데가 있을지 누가 아는가?
4. 그밖에 어떤 점에서 재미를 느껴야 할지 토론해보자.