TEX Help on Fundamentals

gromov*

차 례

*http://faq.ktug.org/faq/gromov/TeXHelpOnFundamentals.

제 1 절 TEX 기본 개념

이 페이지는 TEX을 사용하는데 있어서 가장 기본이 되는 사항을 모아놓는다. 많은 사람들이 TEX을 사용하다가 질문하는 것을 보면 TEX이 어떤 사고방식으로 만들어졌는가를 잘 몰라서 질문하는 것들이 있다. 자신에게 필요한 것만 알면 된다고 할지도 모르지만 TEX이 아무리 정교한 시스템이더라도 결국은 컴퓨터의 일이고 아무리 의미와 상관있는 이름을 명령으로 사용한다고 하여도 조금 테크니컬해지면 의미와 상관없는 조합을 사용하거나 해야 하므로 TEX의 기본 철학을 이해하는 것이 무엇보다도 중요하다.

이 부분은 아마도 Knuth의 *The TEXBook*과 Spivak의 *The Joy of TEX*을 읽어보아야 하겠지만 꼭 필요한 것 조금만 써 두도록 하자.

1.1 TEX을 이해하는 방법

1. TEX은 DTP (DeskTop Processing) 프로그램이다. 즉 책을 출판할 수 있는 수준의 output을 만들어주는 프로그래밍 언어이다.

   - 따라서 이것은 WYSIWYG 스타일의 워드 프로세서와는 다른 것이다.
   - 그러나 이것은 다른 DTP 프로그램들과는 다르게 조판하는 수고를 최소화할 수 있도록 고안된 DTP 프로그램이다. 즉, 대부분의 조판은 자동으로 되어서 글을 쓰는 사람들이 이 부분을 잘 몰라도 할 수 있게 만든 것이다. 그렇다고 해서 진짜로 모르고 할 수는 없겠지만 보편적인 포맷을 따른다면 많은 부분을 신경쓰지 않아도 되게 되어 있다.

2. 이것은 프로그래밍 언어이므로 언어를 알아야 한다. 자신이 사용하는 많은(그러나 상대적으로 적은) 명령어들을 익히고 사용할 수 있어야 한다.

3. 많은 명령어들은 영어로 읽는 방법에 가장 가깝게 정의되었다. 특히 수식과 관련된 명령어들은 영어로 수식을 읽는 방법에 많이 가깝다. 따라서 그냥 외우기 보다는 영어로 수식을 어떻게 읽는가를 공부하면서 익힌다.

1.2 TEX의 장단점

TEX은 많은 명령어를 사용하는 프로그래밍 언어라는 특징이 있고 이것은 장점이면서 단점이기도 하다.

단점

1. 배우는 그자리에서 한 문장이라도 제대로 만드는데 워드프로세서보다 시간이 걸린다.

2. 제대로 output을 만들어 냈어도 어떻게 한 것인지 잘 이해되지 않는다.

3. 워드와는 달리 자신이 잘못 입력을 했을 때 잘못한 점을 금방 이해할 수 없으며, 잘못된 부분을 찾는 것도 쉽지 않다.

4. 특별히 어떤 모양을 만들고 싶을 때 다른 사람의 설명을 들어도 워드보다 이해가 쉽지 않다.

5. 원하는 font를 사용하고 싶을 때 font만 가지고 있으면 별 도움이 안 된다.

장점

1. 하안글이나 MS워드처럼 예전에 만든 파일을 나중에 보려고 하면 제대로 보이지 않는다든가 수식의 포맷이 모두 바뀌어서 나중에 일일이 고쳐준다거나 하는 일이 거의 없다.

    - 특히 본문의 글자 크기를 전체적으로 키워줄 때 수식의 글자 크기도 자동으로 커진다. 워드프로세서처럼 일일이 수식의 글자 크기를 고쳐주거나 할 필요가 없다.

2. 아티클이나 책을 다 만들 때쯤 갑자기 수식을 하나 더 집어넣거나 참고문헌을 하나 더 넣어도 책을 모두 읽으며 참고문헌 번호나 수식 번호를 다 고쳐줄 필요가 없다.

3. 프로그램의 완성도가 높아서 에러때문에 업그레이드하는 식의 문제는 거의 없다.

4. MS Windows, Mac, Linux, Unix 어디서 사용해도 똑같은 파일을 가지고 사용할 수 있다.

5. 여러 사람의 작업을 모을 때, 장, 절이나, 수식의 번호, 그림, 표의 번호 등을 고칠 필요가 없다. 특히 여러 사람이 만든 포맷이더라도 결과는 똑같이 나온다.

6. 일반 워드프로세서를 사용할 때 고민하면서 내가 조판한 결과보다 아무 생각 없이 만든 TEX output이 훨씬 더 보기 좋다.

7. 일반 워드프로세서보다 만들 수 있는 것이 훨씬 더 많다.

1.3 TEX을 사용하는 마음가짐

1. TEX은 언어이다. 언어를 배우는 것이 하루아침에 되지는 않는다. 그러나 TEX의 장점은 기본 Template만 가지고도 (`\section` 등만 사용하여) 훌륭한 문서가 만들어진다는 것이다. 나머지 필요한 것들은 시간을 두고 익힌다. (표, 그림, 수식에는 시간을 많이 투자하여야 한다. 그러나 하안글이나 워드 같은 것에 들이는 시간에 비하면 훨씬 보상이 크다.)

2. TEX을 사용하려면 다른 워드프로세서에서 shortcut key를 외우는 것처럼 명령어와 그 syntax에 시간을 투자하여야 한다. (실제로 사용하는 명령어는 그리 많지 않다. 그리고 기호 등 많은 명령어는 읽는 방법 그대로여서 따로 외우지 않아도 된다.)

3. 마우스를 사용하지 않으려는 것이 중요하다. 많은 경우 명령어의 입력으로 해결되므로 키보드로 입력하는 것에 익숙하여야 한다. 될 수 있으면 한글과 영어의 타자 연습을 해 두는 것이 좋다.

4. 프로그램 언어이므로 에러가 발생했을 때 debug하려는 마음가짐을 가지고 있는 것이 중요하다. 에러를 무서워하면 안 된다. (실제로 발생하는 에러는 몇 종류 없다. 조금 익숙해지면 대부분의 에러를 자기 혼자 해결할 수 있다.)

5. KTUG 게시판과 같이 여러 사람이 질문하고 답한 곳을 뒤져보는 노력을 한다. search 메뉴를 사용하여 대부분의 설명을 곧바로 찾을 수 있다. 실제로 이곳을 이용하는 것이 웬만한 설명서와 help menu를 이용하는 것보다 훨씬 빠르고 정확하다.

제 2 절 TEX의 구성

2.1 TEX Engine

TEX은 WISIWYG 형태의 워드프로세서가 아니다. 이것은 몇 안되는 프로그래밍 언어로 출판을 위한 조판을 해 주는 시스템이다.

이것은 입력된 소스를 일정한 형태의 출력물로 번역(컴파일)하는 역할을 하는 핵심 프로그램을 포함하고 있다. 이 핵심 프로그램을 "TEX Engine"이라고 부른다.

한편, 흔히 format이라고 부르는 "사전에 정의된 명령어(macro) 집합"은 일종의 라이브러리와 같은 역할을 한다. 가장 잘 알려진 format이 LATEX이다. 이밖에도 ConTEXt나 plain TEX도 format에 해당한다.

사용자는 TEX Engine과 format이 결합된 형태로 명령행에서 실행하게 된다. 예를 들어 `latex`이라는 명령을 실행하게 되면 이것은 pdfTEX 엔진으로 LATEX format을 불러들여 dvi mode로 입력 파일을 처리하게 한다는 의미이다.

이 내용을 표로 정리하면 표 1과 같다. 보통 XETEX, LuaTEX 엔진과 구별하여 pdfTEX, ε-TEX을 "레거시 텍"이라고 부르는데, 이 "레거시 텍"과 다른 두 엔진의 (사용자 입장에서) 가장 큰 차이는 폰트 사용 방법에 있다. 즉 레거시 텍은 전통적인 tfm 방식의 폰트만을 처리할 수 있는 반면, 다른 두 엔진은 트루타입, 오픈타입 폰트를 바로 사용할 수 있다는 점이다.

어떤 TEX 엔진을 사용하는가? 이것은 필요에 따라 다르다고 할 수 있다. 그러나 2013년도 말 현재 우리나라에서 한글을 사용한 TEX 조판을 한다면 거의 99% XELATEX을 쓰는 것이 좋다. 한글 사용 패키지 ko.TEX은 XETEX 엔진에서 잘 동작한다.

표 1: 엔진, 포맷

실행 명령	엔진 프로그램	포맷	출력 모드 (디폴트)
tex	TEX	plain TEX	dvi
etex	ε−TEX	plain TEX	dvi
pdftex	pdfTEX	plain TEX	pdf
latex	pdfTEX	LATEX	dvi
pdflatex	pdfTEX	LATEX	pdf
xetex	XƎTEX	plain TEX	pdf
xelatex	XƎTEX	LATEX	pdf
luatex	LuaTEX	plain TEX	pdf
lualatex	LuaTEX	LATEX	pdf
context	LuaTEX	ConTEXt	pdf
texexec	pdfTEX, XƎTEX	ConTEXt	pdf

2.2 Macros

TEX의 특징은 문서의 스타일에서부터 자잘한 기호 까지도 모두 기본 엔진은 가지고 있지 않다. 이런 것을 정의하고 사용할 수 있도록 하는 것은 TEX 시스템의 거의 대부분을 차지하는 매크로(Macro) 파일들이 하고 있다. 이 매크로 파일들은 누구나 정의할 수 있으므로 TEX의 막강한 힘이 되고 있다. 만들어진 많은 매크로들 가운데 활용도가 인정된 것들은 선별되어서 TEX 시스템에 추가된다.

매크로에는 몇 가지 레벨이 있다.

Format

매크로 가운데 TEX Engine의 바로 위에서 조판의 기본적인 부분을 전부 담당해서 그 문서의 스타일을 개략적으로 정해주는 부분은 가장 중요한 것으로 plainTEX, LATEX과 같은 것이 있다. 이것은 워낙 큰 파일이어서 문서에서 그냥 부르면 시간이 오래 걸리므로 미리 기계어로 바꾸어 컴파일해 두었다가 필요할 때 불러서 쓴다. 이렇게 컴파일 된 LATEX은 `latex.fmt`라는 이름의 binary인 format (`fmt`) 파일이 되어 있으며 우리가 latex이라는 명령어로 컴파일 명령을 내리면 TEX이 이 format file을 불러 (재빨리) 읽고 우리 문서를 조판한다. 이 fmt 파일은 요즘은 사용자가 별로 생각할 부분이 없으며 거의 TEX Engine 자체라고 생각되고 있다.

Class와 Style

예를 들어 LATEX 포맷을 사용할 때 많은 경우에 기본 LATEX이 제공하는 것보다 더 자세한 문단 스타일이나 기호들을 사용할 수 있게 하려고 한다면, 이에 추가하여 정의한 내용들을 한 파일로 해서 문서보다 먼저 읽도록 하면 좋다. 이런 파일을 보통은 style file이라고 하고 `filename.sty` 꼴의 이름을 붙인다.

자세한 사항은 다음을 참조한다. 클래스와 스타일.

Class 그런데 이런 것들 가운데도 많은 사람들이 공통으로 사용하는 큰 style file은 class라는 이름으로 정의되어 문서 서두에

```
\documentclass[options]{classname}
```

라는 명령으로 불려진다. 이런 class 가운데 가장 유명한 것은 `article.cls`이며 아마도 가장 많이 쓰일 것이다. 이 밖에 수학계에서는 AMS (American Math. Soc.)가 만든 `amsart.cls`가 잘 쓰이고, 다른 학회나 논문지 등도 자신들의 필요에 맞는 class file들을 만들어 배포하고 있다.

대부분의 경우에 이러한 파일을 사용하면 많은 수고를 덜 수 있으므로 한 두개 정도의 class 파일 안에 정의된 명령어들을 공부해 두는 것이 좋다.

Style File 이러한 class를 사용하면서도 더 필요한 특수한 기능이나 내가 많이 쓰는 명령어를 축약시켜 놓은 것 등을 따로 저장해 두고 사용하는 비교적 작은 사이즈의 명령어 모음 파일을 style 파일이라고 부르며 `filename.sty` 꼴의 이름을 붙인다.

이런 파일들은

```
\usepackage[options]{packagename}
```

와 같은 식으로 부르게 된다. 한 파일에서 여러 개의 style file을 부를 수 있으며 이는 그 내용을 그대로 순서대로 파일 안에 써 놓은 것과 같다고 생각해도 된다.

많은 style file이 이미 TEX 시스템에 들어 있으며 개인적인 파일들을 만들어 `texmf-local`과 같은 directory 아래에 TEX의 directory 구조에 맞게 넣어 놓거나 또는 파일을 컴파일하는 현재 directory에 넣어 두면 사용할 수 있다.

이러한 명령어 모음 파일은 `filename.tex`이라 명명하고

```
\input{filename}
```

과 같이 불러도 된다.

이러한 모든 것들은 LATEX이라면

```
\begin{document}
```

명령보다 앞쪽인 preamble에 적혀 있어야 대부분의 경우 문제가 없다.

2.3 폰트 사용에 관한 이해

레거시 텍의 Font와 TFMFiles

TEX은 조판을 위한 글꼴이 복잡하다. TEX은 자체로 사용하는 글꼴이 따로 있으며 이는 처음에 만들어질 때 MetaFont라는 형식으로 만들어졌다. 이는 최근의 TrueType, Vector

Font 등의 원조 정도라고 이해된다. 각각의 글꼴을 글꼴 모양의 경계 곡선을 정의한 것이라고 보인다. 이것을 사용할 때에 bitmap 그림인 pk font file로 변형하여 사용하고 사용이 끝나고 일정 기간이 지나면 지워버리는 식으로 사용하였다.

이것은 초기 TEX이 개발되었을 시절의 PC가 하드디스크 10MB, 또는 이것도 없이 500KB 플로피디스크 2장을 사용하고 있었다는 것을 생각하면 이해가 간다. (80년대 중후반 당시 10MB HDD 의 가격이 10만원이었다.)

지금도 TEX은 이 시스템을 견지하고 있다. 이 구식이 다 되어 가는 시스템을 이해하려면 우선 TEX이 어떻게 조판하여 우리에게 보여주는가를 알아야 한다. 기본적인 TEX은 파일을 우리에게 보여줄 때 다음과 같은 과정을 거친다.

source file → dvi file → ps 또는 pdf file

요즈음은 dvi file을 거치지 않고도 pdf file을 만들어주는 pdfTEX이 보편화됐지만 그래도 dvi file의 잔재는 글꼴에 남아 있다.

DviFile

dvi란 말은 *D*evice *I*ndependent라는 말을 줄인 것이다. 여러 컴퓨터 시스템과 여러 OS에서 같은 일을 하게 하면서도 작은 컴퓨터 자원을 효율적으로 쓰려고 TEX은 조판할 때 Font를 다 읽어보지 않는다. Font는 너무 커서 모두 메모리에 올리는 것도 힘들고 하므로 Font에서 조판할 때 필요한 기본 정보인 글자의 폭, 높이, 앞뒤의 간격, 글자가 줄 baseline과 갖는 위치 등등의 수치적 정보만 뽑아서 읽는다. 매번 뽑는 것도 시간이 많이 걸리므로 글꼴에서 이런 정보만 뽑아 놓은 아주 작은 파일들을 tfm file이라는 이름으로 만들어둔다. (tfm은 tex font metric이라는 말의 줄임말이다.)

한번 소스를 읽어들이면 한줄한줄을 조판할 때는 tfm 파일의 내용만을 가지고 조판하여 글자 자체는 없이 빈칸만으로 페이지를 만든다. 이에 대한 자세한 내용은 *TEXBook*등을 참조하면 좋다. 이렇게 빈 칸과 그곳에는 어떤 글꼴의 글자가 들어올 것인지만 적어놓은 파일이 dvi 파일이다. 이것만 있으면 font 자체가 안 들어있는 dvi file이지만 TEX이 깔려있는 시스템 어디를 가지고 가도 그곳의 font를 가지고 출력할 수 있게 된다. 따라서 dvi 파일은 사이즈가 상당히 작다.

이제 이것을 font가 없는 곳에서도 사용가능하게 하려면 여기다 실제로 font 그림을 모두 심어서 만든 파일이 필요하고 이것들이 ps 또는 pdf 등등의 파일들이다. 이것들은 글꼴 그림이 들어가 있는 관계로 그 사이즈가 매우 크다. 지금은 font 크기는 별로 문제가 되지 않으니까 dvi file을 쓸 필요는 거의 없지만 아직도 기본적으로 글꼴은 pk와 tfm으로 나뉘어 있는 것이다. 그리고 아직도 dvi를 사용하는 사람들이 있다.

트루타입과 오픈타입

최근에 XeTEX, LuaTEX이 개발되어 일반 Font를 사용할 수 있게 진화하고 있다. 이 일반 폰트는 시스템에서 사용하는 폰트들이다. 트루타입과 오픈타입이 대표적이며 윈도우즈 시스템의 경우 `C:\Windows\Fonts` 폴더에 들어 있는 많은 `ttf` 확장자를 가진 폰트들이 여기에 해당한다.

이 폰트들을 잘 사용하려면 폰트의 이름을 기억하고 XeTEX이나 LuaTEX에서 이 폰트들을 부르는 방식을 알아두어야 한다. fontspec이라는 패키지를 이용하여 LATEX에서는 이것을 손쉽게 할 수 있게 되어 있다. 자세한 사항은 XeTEX-ko나 LuaTEX-ko 매뉴얼을 참조하여야 한다.

2.4 기타

(이 절의 내용은 비어 있습니다.)

제 3 절 TEX 활용을 위한 구성

3.1 TEX Distribution Files

한 시스템에서 TEX을 사용하려면 보통 단 한 개의 TEX Distribution을 설치한다. 자신이 TEX 시스템을 구성하여도 안 되는 것은 아니지만 노력이 많이 들므로 일반적으로 만들어져 있는 Distribution을 전부 또는 일부 설치하여 쓴다. 앞의 설명을 보고 자신에게 맞는 Distribution을 고른다.

최근에는 MikTEX과 TEX Live 두 개가 주종을 이루고 있다. KTUG에서 제공하는 ko.TEX Live라는 설치 프로그램은 TEX Live를 설치해준다.

MS Windows라면 단 한 개의 Distribution 밖에는 설치할 수가 없다. 아마 Linux도 그렇지 싶다.(예전에는 물론 그랬었다.)

3.2 에디터

TEX Distribution이 기본적인 editor를 제공해 준다. 지금은 어떤 TEX Distribution이라도 TEXworks라는 에디터를 바로 사용할 수 있다. TEXworks는 훌륭한 에디터지만 사람들은 editor를 따로 설치해서 쓰는 경우도 많다. 이것은 KTUG 등에 잘 설명되어 있으며 어떤 것을 사용할지 어떤 장점이 있는지 등을 살펴보고 결정해야 한다.

익숙한 사람들은 Emacs 계열을 잘 사용하지만 윈도우창과 버튼에 익숙한 초심자들에게는 무리다. 보통은 윈도우에서 잘 돌아가는 몇 가지 Free 또는 Shareware를 선택한다.

에디터는 자체적으로 기본 코드만을 입력해주는 것을 사용해야 한다. 에디터가 문단 format을 한다던가 하는 것은 내가 입력한 글자 외에 다른 기호를 나 모르게 파일에 넣는다는 뜻이므로 TEX용으로는 부적합하다. 대부분의 프로그래밍용 에디터는 다 쓸 수 있다.

한 가지 주의할 것은 최근의 TEX의 발전 방향에 따라서 UTF-8 등의 unicode를 제대로 다루는 에디터일 필요가 있다는 것이다. KTUG 홈페이지에서 자세한 사항들을 읽어볼 것.

3.3 Viewer

조판이 끝난 문서를 내가 보고 인쇄할 수 있게 해 주는 프로그램이 viewer이다. 예전에는 dvi 뷰어(yap, dviout 등)가 필요했지만 지금은 pdf가 기본 출력이나 마찬가지이므로 뷰어는 pdf 뷰어면 충분하다.

TEXworks 에디터에는 pdf 뷰어가 딸려 있으므로 이것을 이용해도 좋고, OS별로 pdf를 읽는 프로그램을 뷰어로 사용해도 좋다. 보통 Adobe Reader를 사용한다. 그러나 이 밖의 것을 사용하는 경우도 많다. 자신이 따로 설치하고 설정해서 사용하는 것도 가능하다. 이에 대한 부분도 KTUG의 홈페이지를 참조할 것.

3.4 그림 그리는 도구

문서를 작성하면서 가장 복잡한 부분이 그림과 관련된 부분이다. 깨끗한 그림을 그려 넣는 것은 매우 어렵다. 보통 몇 가지 방법을 사용한다. KTUG Faq 페이지에 자세한 설명이 있다.

1. 이미 그려 있는 그림을 picture format (jpg, png, pdf, eps)으로 삽입하는 방법이다. graphicx 패키지와 `\includegraphics` 명령을 사용한다. 손으로 그려 scan하거나, 다른 그림 그리는 프로그램을 활용한다.

2. LATEX 등의 기본 명령어를 이용하여 문서에서 직접 그림을 그린다. 많은 명령어를 공부하여야 하고 쉽지 않다. 그림의 복잡성도 한계가 있다. 그림은 pallette에서 그리고 이를 LATEX 명령으로 번역해주는 프로그램도 있다.

3. 훨씬 다양한 그림을 coding할 수 있게 해 주는 것들로 TEX macro package들이 여럿 있다. 대표적으로 PSTricks나 XYPic, Metapost 등을 들 수 있다. 사용법을 많이 익혀야 한다. 이에 대한 것은 *LATEX Graphics Companion*이란 책에 잘 설명되어 있다.

제 4 절 TEX 문서의 구성

TEX 문서는 다음과 같은 순서로 이루어져 있다.

1. Preamble

2. 본문

4.1 Preamble

이 부분은 본문을 식자하는 방법이나 거기서 쓰이는 기호를 정의해 두는 부분이다. 대부분의 워드는 이 부분을 설정하는 많은 메뉴를 가지고 있다. 그러나 TEX은 이것을 직접 입력한다.

맨 앞에는 document class를 설정하면서 이의 option까지 지정한다.

그 다음에는 내 문서 조판에 필요한 많은 다른 style file등을 적어 둔다. `\usepackage` 명령을 사용한다. 여기서도 option을 같이 지정한다. 이 부분에 들어가는 것은 본문에 사용하는 글꼴의 변경, 페이지 크기 조정, 사용할 기호와 특수한 명령어 관련된 style, 특별한 문서 기능을 위한 것 (hyperref 같은 것) 등등이 있다.

4.2 본문

LATEX을 사용하면 본문은

```
\begin{document}
```

와

```
\end{document}
```

사이에 들어가게 된다. 그리고 `\end{document}` 다음에 적힌 모든 것은 TEX이 읽지 않는다.

또 본문에 적혀있더라도 각 줄에서 % 글자가 나온 이후의 글자들은 읽지 않는다. 즉 % 는 커멘트를 적는 기호이다. % 기호는 그 줄에만 효력이 있으며 Return키가 한번 들어가면 그 효력은 없다. TEX에는 어디에서 어디까지는 comment이다 라는 식의 명령어는 없다.

이 % 기호를 잘 활용하면 매우 편리하다. 여러 가지를 적어 놓아도 그 가운데 필요한 것만 남기고는 모두 앞에 %를 붙여두면 같은 파일을 여러 용도로 사용할 수 있다.

4.3 small2e.tex

LATEX의 첫 개발자인 Leslie Lamport가 작성한 작은 샘플 파일 `small2e.tex`을 한글 환경에 맞게 조금 수정하면 다음에 제시하는 것과 같은 모양이 된다. 이 샘플을 잘 살펴보는 것으로 자신의 첫번째 LATEX 문서를 작성할 수 있을 것이다.

```
% This is a small sample LaTeX input file (Version of 10 April 1994)
%
% Use this file as a model for making your own LaTeX input file.
% Everything to the right of a  %  is a remark to you and is ignored by LaTeX.

% The Local Guide tells how to run LaTeX.

% WARNING!  Do not type any of the following 10 characters except as directed:
```

```
%               &   $   #   %   _   {   }   ^   ~   \

\documentclass{article}        % Your input file must contain these two lines
\usepackage{kotex}             % 한글 사용
\begin{document}               % plus the \end{document} command at the end.


\section{Simple Text}          % This command makes a section title.

Words are separated by one or more spaces.  Paragraphs are separated by
one or more blank lines.  The output is not affected by adding extra
spaces or extra blank lines to the input file.

단어 사이는 스페이스로 분리된다. 스페이스는 몇 개라도 상관없다.
문단은 빈 줄로 분리된다. 빈 줄이 몇 개라도 상관없다.

Double quotes are typed like this: ``quoted text''.
Single quotes are typed like this: `single-quoted text'.

Long dashes are typed as three dash characters---like this.

Emphasized text is typed like this: \emph{this is emphasized}.
Bold       text is typed like this: \textbf{this is bold}.

\subsection{A Warning or Two}  % This command makes a subsection title.

If you get too much space after a mid-sentence period---abbreviations
like etc.\ are the common culprits)---then type a backslash followed by
a space after the period, as in this sentence.

Remember, don't type the 10 special characters (such as dollar sign and
backslash) except as directed!  The following seven are printed by
typing a backslash in front of them:  \$  \&  \#  \%  \_  \{  and  \}.
The manual tells how to make other symbols.

\end{document}                 % The input file ends with this command.
```

4.4 수식

문장 속에 들어가는 in-line 수학식은 \(로 시작하고 \)로 끝난다. 종래에는 $ 표시로 시작과 끝을 같이 표시했지만 아주 짧은 a 같은 것은 이것이 편리해도 조금만 길어지면 시작과 끝을 잘 알 수 없어서 오류가 발생할 가능성이 높아지므로 \(x+y\) 이런 식으로 표기하는 것이 좋다.

문장 사이에 문단처럼 들어가는 수학식은 \[로 시작하고 \]로 끝난다. 이것을 display

math라고 하는데 실제 수학식을 표현하기 위해서는 이것만으로는 불충분하고 amsmath 패키지가 제공하는 여러 환경을 적절히 사용하는 것이 필요하다.

예전에 많이 쓰이던 여러 줄 수식 eqnarray는 더이상 사용하지 않는 것이 좋으며 amsmath의 split, align 등의 환경을 써서 입력해야 한다.

수학식 입력에 필요한 사항을 다른 글에서 밝혀둔 것[1]이 있다.

[1] http://faq.ktug.org/faq/gromov?action=download&value=tex_u_ywk_xe.pdf