스크리브너와 라텍

Nova De Hi

2014년 2월 15일

요 약

이 문서는 2014년 1월–2월 동안 고려대학교에서 진행된 「스크리브너와 라텍 활용」 강좌의 내용을 요약한 것입니다. 공부의 기회를 주신 김영욱 교수님, 그리고 촬영 같은 궂은 일을 마다 않으신 이상욱 고영미 교수님, 강의 진행의 실무를 맡아 고생하신 김소영 선생께 감사드리며, 강좌에 참석하여 대화를 나누었던 모든 분들께도 감사 말씀을 드립니다. 이 글에서는 일부 못다 말한 내용을 추가하면서 강좌의 의도와 내용 전체를 알기 쉽게 적어보려 하였습니다. 더불어, 이 문서 자체가 그간의 강좌 전체를 요약한 샘플 문서가 될 수 있을 것입니다.

이 글은 Scrivener의 사용설명서가 아닙니다. 스크리브너의 여러 매력적인 기능 중에서 LaTeX과 관련된 것만 설명하였고 그것도 기능 중심이 아니라 "글쓰기" 라는 목적에 비추어서 설명하였기 때문입니다.

[1] http://creativecommons.org/licenses/by-nd/2.0/kr/

차 례

CHAPTER 1

컴퓨터로 글쓰기와 라텍

컴퓨터로 글쓰기가 당연시되는 오늘날 워드 프로세서는 가장 중요한 컴퓨터 활용 방법의 하나가 되었다. 그러나 워드 프로세서라는 도구는 의외로 글쓰기와 생각의 흐름을 방해하는 경우가 없지 않다. 그래서 많은 사람들이, "저술자는 글의 내용에만 신경을 쓰라"고 권하고 그 한 가지 방법으로 LaTeX과 같은 도구를 추천하는 것이다 [1].

1.1 LaTeX의 장단점

LaTeX의 장단점을 "저술 도구"라는 관점에서 살펴보자.

1. 수학식을 기술하는 데 있어서 필요불가결하다. 이것은 *현실적으로* 다른 어떤 것도 대체하지 못한다.[1]
2. 매우 높은 수준의 조판 도구로서 디자인을 매우 충실하게 구현할 수 있다. 따라서 이것은 저작 도구인 동시에 "출판 도구"이기도 하다.
3. 문서를 논리적이고 구조적으로 작성할 수 있게 한다.

그러나,

1. LaTeX 마크업 자체가 어렵다. 라텍 마크업은 수식 표현을 위하여 표준화된 마크업이 잘 설계되어 있는 편이지만 조판과 출판 디자인 목적의 마크업은 매우 판독하기 어렵고 디버깅이 쉽지 않은 언어인 TeX을 사용하고 있다. 그리고 이 서로 다른 목적의 마크업들이 명확히 구분되어 있지 않아서 혼란

[1] 혹 HWP가 수식 표현 도구로서 제 구실을 하고 있지 않으냐는 반론이 있으나, HWP 수식은 표준에도 맞지 않으며 재사용 가능성이 없는 것으로 그런 것을 "문서"라고 말하기 어렵다.

스러운 경우가 종종 있다. LaTeX 초심자들이 특히 당황하는 것이 이런 점에 적응하는 데 시간이 걸리기 때문일 것이다.

2. 오늘날 TeX 사용 환경은 대부분 최종 출력물인 PDF 파일을 생성하는 것이 너무 쉬워서 오히려 지나치게 자주 최종 결과물을 참조하는 폐단이 있다. 그러다보면 끊임없이 "여백과 폰트"에 신경을 쓰게 되고 이 마크업 언어의 원래 의도에 어긋나게 오히려 글쓰기의 주의를 분산시키게 되는 것이다. 그 폐단이 워드 프로세서의 그것만큼 크다고 할 수는 없어도 LaTeX 피로도가 높은 것도 사실이라고 본다.

1.2 형식과 내용의 분리

LaTeX이 금과옥조로 삼는 슬로건, "형식과 내용의 분리"라는 점을 조금 더 언급하자.

원래 이 슬로건은 워드 프로세서에 대한 LaTeX의 우위를 강조하려는 데서 나온 것이다. 워드 프로세서의 사용자 경험은 그 응용 프로그램의 디자인상 format에 해당하는 속성과 contents에 해당하는 속성을 구분할 수 없는 반면, 라텍과 같은 "마크업 방식"의 글쓰기는 이 둘이 control sequence와 내용으로 선명하게 구분되는 데서 왔다고 할 수 있다. 즉, `\section{섹션}`과 같은 마크업의 "의미론적(semantic)" 특성과, 그것이 실제 출력물이 되었을 때의 구체적이고 개별적인 "모양"은 구분될 수 있고 구분되어야 한다는 것이다.

웹-기반의 인터넷 생활을 하고 있는 오늘날, 이 철학은 의외로 가까운 곳에서 잘 구현되어 있다. 그것이 바로 HTML과 CSS의 분리를 통한 웹사이트의 설계이다. 그림 1.1과 그림 1.2는 지금 작성 중인 이 문서를 HTML preview하고 있는 장면인데 완전히 똑같은 내용에 대하여 서로 다른 CSS를 적용한 결과이다. 라텍에서 내용과 형식의 분리란 본질적으로 이와 동일한 것이다.

1.3 마크업

문서 생산자에게 있어 형식과 내용의 분리는 그 역사적 연원이 꽤 오래 되었다. 예전의 문필가들은 원고를 출판업자에게 보내면서 "Script(원고)"에 수많은 출판상의 고려 사항에 대한 주석을 붙였다. 이것은 배행간으로 타자친 원고의 행간과

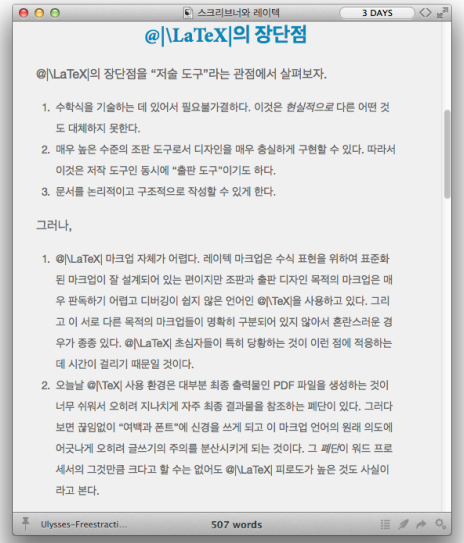

그림 1.1: 예제그림1

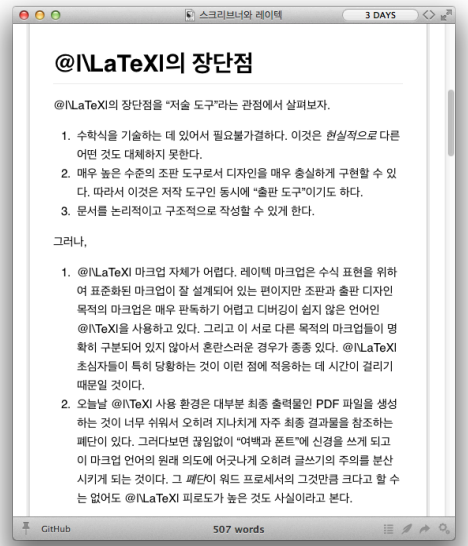

그림 1.2: 예제그림2

여백에 들어갔다. 그리고 편집자와 조판 담당자가 또 많은 주석을 붙이면서 책을 만들어갔던 것이다.

오늘날은 이 작업을 *mark-up*이라는 것으로 수행한다. 본문의 내용을 온전히 보존하면서 그것을 어떤 다른 목적으로 활용(여기서는 출력물의 포매팅)하기 위하여 필요한 meta 정보들을 체계적으로 기록하기 위해 개발된 것이 XML이고 HTML은 그 일부이다.

그러나 대부분의 mark-up language들은 인간-친화적이라기보다는 기계-친화적이다. 저술 활동만을 문제삼을 때 XML이나 HTML은 너무 번거롭고 사람이 직접 적어넣을 만한 것이 못된다. 차라리 LaTeX 마크업이 낫다. 그럼에도 불구하고 출판(온라인이든 오프라인이든)을 목적으로 하는 문서의 작성에 있어서 **분업**, 즉 저자는 contents 부분을 전담하고 포매터는 포매팅 부분을 전담해야 한다는 사실만은 변함이 없으며, 그것이 바로 LaTeX의 장점이라고 주장해왔던 바로 그 사실이라는 점을 확인해두기로 하자.

결론은 이렇다. 저자가 저술 활동을 하는 동안, 문서의 모양과 포맷에 대해서 **신경쓰지 않게** 해주는 것이 가장 훌륭한 저작도구라는 것.

이 관점에서 LaTeX이 지나치게 번거로운 점이 있고 "저자의 주의를 다른 곳으로 돌리게 만드는" 요소가 남아 있다는 점을 언급하였다. 그러나 출력물의 품위와 그 활용가능성을 생각할 때 LaTeX을 버릴 수는 없다. 본질적으로 라텍을 활용하게 하면서도 저자의 저술 과정 동안은 번쇄한 라텍 마크업을 잠시 잊거나 최소한으로 활용하게 해준다면 이것이 하나의 해결책이 되지 않겠는가?

이런 점에서 라텍의 대안적 활용 방안을 생각해보고자 한다.

1.4 라텍의 활용 방법: 제안

핵심만을 말한다면, "라텍을 사용하되, 라텍 '날코딩' (명령과 환경의 마크업)을 되도록 하지 않는다"는 전략이다. 저자는 **최소한의** 마크업만으로 문서를 작성하고, 그 문서의 포매팅은 라텍에게 맡기되, 원고를 "출력 가능한" 라텍으로 번역하는 일은 컴퓨터로 하여금 하게 한다.

이 전략을 채택하면 복잡하고 거슬리는 포매팅 명령을 신경쓰지 않고 말 그대로 문서 작성에 전념할 수 있을 것으로 기대한다. 쉽고 간단한 몇 가지 약속만으로 문서를 작성하게 될 것이므로 "복잡한 포매팅"에 대한 욕구나 "신경쓰이는 overfull"을 잠시 잊을 수 있다. 이러한 것은 라텍으로 최종 출판물을 만드는 과정에서

굳이 필요하다면 하면 되는 것이고, 이상적인 것은 그마저 하지 않더라도 훌륭한 출력물을 얻을 수 있으면 가장 좋은 것이다.

이에 더하여 라텍이 가지는 장점, 즉 심지어 TEX이 설치되어 있지 않은 곳에서도 원고의 내용은 파악할 수 있다는 바로 그 "플레인 텍스트"의 장점이 지금 제안하는 이 방법에서는 훨씬 더 명료하게 두드러지게 된다는 점을 부기해두겠다.

우리가 도입하려 하는 대안은 MultiMarkdown이다. 이것은 원래 HTML 마크업을 간소한 몇 가지 규약만으로 작성하기 위해 John Gruber가 만든 Markdown이라는 "lightweight mark-up language"를[2] 확장한 것인데 그 가운데 특히 LATEX을 지원하는 기능이 추가되고 강화되었으며 version 4에 이른 지금 거의 문제없이 사용해볼 만하다고 판단하여 소개하려 한다.

MultiMarkdown의 저자인 Fletcher Penney의 홈페이지에 있는 "멀티마크다운이란 무엇인가?" 부분을 우리말로 옮긴 자료도 있다 [3].

이를 통하여 저자는 쉽고 편안하게 문서를 작성하고, 더불어 강력한 라텍의 조판 기능의 도움을 얻어 원하는 대로의 포맷을 가지는 멋진 문서를 생산할 수 있게 될 것을 기대하는 것이다.

[2]인간이 읽고 쓸 수 있는 간소한 마크업 언어인 소위 "경량화 마크업 언어"는 그 종류가 엄청나게 많다. 그 가운데 가장 성공적인 것 하나가 Markdown이다. [2]를 참고.

CHAPTER 2

소개와 준비

2.1 스크리브너라는 앱

작가나 과학자와 같이 전문적으로 글을 쓰는 사람을 위한 매킨토시용 애플리케이션으로 Scrivener가 있다.[1]

그림 2.1: Scrivener

"글쓰기"에 특화된 앱으로, 짧게는 블로그 포스팅에서 장문의 소설, 논문, 드라마 대본에 이르기까지 여러 종류의 글을 쓰는 데 편리한 환경을 제공하여 일찍부터 명성이 높았다. 매킨토시를 사는 이유가 스크리브너를 쓰기 위해서라고

[1] 윈도우즈와 리눅스 버전도 있기는 하지만 매킨토시 버전에 비하여 기능이 많이 취약하여 우리가 원하는 목적대로 쓰기에 불편한 점이 많다.

말하는 사람도 있을 정도로 킬러앱의 지위를 차지하고 있고 앱 스토어에서도 꾸준히 상위의 매출을 보여준다.

흥미로운 것은 특히 자신이 작성한 원고(Script)를 출판사의 편집자가 원하는 형태(옛날의 타자 원고와 거의 같다)로 만들 수 있는 기능이다. 우리나라와는 달리 작가가 먼저 시와 소설의 원고를 편집자에게 보내고 편집자가 검토하여 출판하는 방식이 아직도 유지되고 있는 것을 생각하면 이것은 영어권 작가지망생에게 필수불가결한 앱이라고 해도 과언이 아닐 것이다. 우리나라 현실과는 많이 동떨어지지만.

스크리브너는 만만찮은 가격($45)의 상업용 프로그램이다. 그리고 이를 제대로 사용하려면 특정 OS(매킨토시)가 필요하다. 그렇지만 가격을 지불하고라도 써볼 만한 가치가 있는 앱이라고 생각한다. 제작사 홈페이지 Literature and Latte[2]에서 시험판을 다운로드받을 수 있고 30일간 사용이 가능하다.

2.2 스크리브너의 사용 방식

스크리브너는 두 가지 서로 다른 방식으로 사용할 수 있다.

- **워드 프로세서**: 이 앱 자체가 워드 프로세서이다. 일반적인 워드 프로세서와 똑같이 동작하며 HTML, RTF, Word 문서, OpenOffice Writer 문서, Final Draft 문서, Fountain Screenplay 문서를 다룰 수 있고 PDF로 인쇄할 수 있다. 전자책(.mobi, .epub 등)을 제작하는 기능도 지원한다.
- **멀티마크다운 에디터**: 플레인 텍스트 에디터처럼 동작하여 multimarkdown 파일로 내보내기 하거나, *LaTeX 문서를 생성하거나*, 라텍을 경유하여 PDF를 제작하거나 할 수 있다. 물론 멀티마크다운을 통하여 HTML 문서를 작성하는 것도 가능하다.

예를 들어 보자면, 그림 2.2와 같이 문서작성도구로 활용한다면 이것은 스크리브너를 "워드 프로세서"로서 사용하는 것이다.

그러나 이것은 우리가 원하는 것이 아니다. 왜냐하면 *라텍을 지원하지 않기* 때문이다. 라텍을 지원하지 않는다는 말의 의미는 입력된 시각적 요소, 폰트 크기,

[2] http://literatureandlatte.com

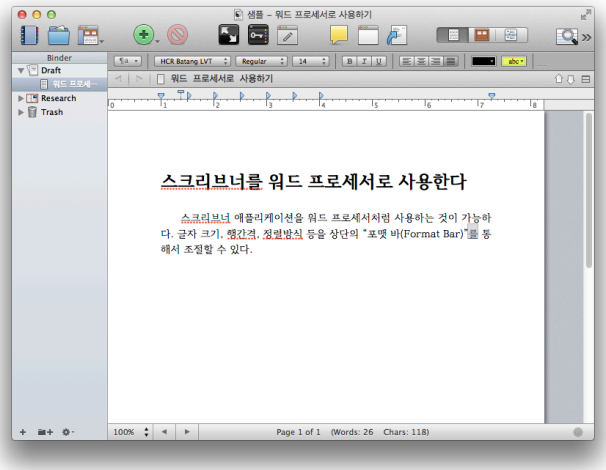

그림 2.2: 워드 프로세서로서의 스크리브너

정렬, 행간격 등이 라텍으로 export되지 않는다는 뜻이다.[3]

따라서 라텍을 겨냥하고 스크리브너를 사용한다면 이러한 시각적 요소를 배제하고 텍스트만으로 문서를 작성해야 한다. 적어도 이러한 요소가 라텍 소스에 아무런 영향을 끼치지 않는다는 점을 이해하고 있어야 한다. 즉, 우리가 활용하려는 스크리브너는 **워드 프로세서가 아니라 에디터**이다.[4]

요컨대, 스크리브너에서 얻을 수 있는 PDF는 두 종류가 있다. 하나는 라텍을 거치지 않고 현재 화면의 모양을 그대로 PDF로 만드는 것이고, 다른 하나는 현재 입력된 텍스트를 (모양은 무시하고) 라텍을 경유하여 만드는 것이다. 이 때 만들어질 PDF의 포매팅을 위하여 멀티마크다운 마크업을 기록한다. 그 결과 현재 화면에서 보이는 모양과는 전혀 다른 라텍 특유의 레이아웃을 가진 문서를 제작하게 된다. 당연히 우리 관심사는 후자이다.

이런 다양한 출력 형식을 지원하기 위하여 스크리브너에는 "컴파일"이라는 특유의 개념이 있다. `File` → `Compile` 메뉴로 접근할 수 있고 Option+Cmd+E

[3]다만 딱 한 가지, 글자의 강조(emph, bold)는 위지윅으로 입력된 것을 markdown으로 변환해준다. 메뉴의 `Format→Convert→Bold and Italics to MultiMarkdown Syntax`. 이마저 자동으로 변환되지는 않는다.

[4]그러나 굳이 라텍 문서로 만들려는 의도가 아니라면 워드 프로세서처럼 사용하는 것도 좋다. 간단한 메일, 블로그 포스팅, 개인적 메모 등을 굳이 라텍 문서로 작성할 필요는 없지 않은가?

단축키로 열 수 있는데, 그림 2.3에서 보는 것과 같은 "컴파일 유형"을 선택할 수 있다. 이 가운데 우리 관심사는 전적으로 맨 마지막의 MultiMarkdown으로 시작하는 몇 개의 선택사항이다. 그 중에서도 `MultiMarkdown → LaTeX`을 중심으로 사용하려 한다. 이 컴파일 옵션을 선택하면 스크리브너 편집창에 시각적으로 입력된 모든 "포맷" 요소는 무시되고 마크업된 것만 출력에 영향을 미친다.

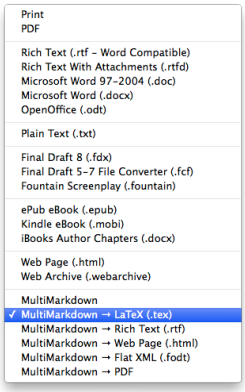

그림 2.3: Compile for…

컴파일 옵션을 설정하는 방법에 대해서는 별도로 다룰 것이다. 여기서 언급하려는 것은 이 "컴파일"이라는 방식이 도입된 결과, 문서 관리에 미치는 영향이다. 스크리브너 파일 그 자체는 문서("프로젝트"라고 한다)를 모아놓은 일종의 데이터베이스이다. 이로부터 mmd도 만들고 pdf도 만들고 tex도 만들고 하여 출력 결과를 얻지만 모든 문서 작성은 스크리브너에서 한다. 실제로 tex이나 pdf나 모두 언제라도 필요하면 생성할 수 있는 것에 지나지 않으므로 원하는 파일을 얻고 나면 지워버려도 상관없다. 이것은 매우 중요한데, tex 파일로 작업하는 경우에 변경사항 추적을 위해서 여러 벌의 복사본을 유지해야 하던 것과 비교해보라. 스크리브너를 통해서 변경사항과 스냅샷을 모두 유지할 수 있다.

또 한 가지, tex 소스로 작업할 때 약간은 성가신 이른바 "예약 문자 제약"이 거의 없다는 점도 특기해야 할 것이다. 편하게 #, &, ^, % 이런 문자를 그대로 입력해도 된다. 물론 멀티마크다운에서 사용하는 몇 가지 문자의 특별한 사용법이 있지만 에러를 토해내거나 하지 않는다.

라텍에 익숙한 분들이 자주 저지르는 실수는 따옴표의 입력이다. 스크리브너에서 따옴표에 grave accent 또는 back quote 또는 quasiquote 문자(`)를 사용하면 안

된다. 이것은 멀티마크다운 예약문자이다. 그냥 키보드의 quote 문자로 따옴표를 열고 닫는 것이 좋다. 겹따옴표는 (back)quote 문자를 두 번 입력하지 말고 보통 워드 프로세서에서 하듯이 Shift-quote로 입력하면 적절하게 열고 닫고 할 수 있다.

2.3 멀티마크다운

멀티마크다운은 마크업 언어를 가리키는 말이기도 하고 마크업된 텍스트를 변환해주는 도구(`multimarkdown`) 이름이기도 하다. 보통 줄여서 mmd라고 한다.

멀티마크다운 자체는 스크리브너와 무관한 툴이다. 스크리브너 없이도 mmd 파일을 작성하여 이를 라텍으로 변환하거나 PDF 출력하거나 하는 일이 가능하다. 스크리브너가 멀티마크다운 작업을 지원하는 것이다.

멀티마크다운은 이름("Multi-")에 걸맞게 마크다운에 비하여 조금 복잡한 규약을 가지고 있다. 그러나 감당이 안 될 정도는 아니고 실제로 알아두어야 할 것은 몇 가지 되지 않는다. MMD Syntax Cheat Sheet[5]을 한 번 읽어보면 mmd로 무엇을 할 수 있는지 잠깐 사이에 훑어볼 수 있을 것이다.

원래 마크다운과 멀티마크다운은 모두 HTML을 겨냥해서 만들어진 것이다. 우리는 지금 라텍과 연결하여 쓰는 문제에 관심을 가지고 있지만 그것은 MMD 자체의 입장에서 보면 부차적이다. 요컨대 MMD가 라텍을 지원하기는 하지만 라텍만을 위하여 만들어진 전용 언어는 아니라는 것. 예를 들어 라텍을 백엔드로 사용하는 입장에서 웹사이트의 URL을 통하여 그림을 문서에 삽입하는 것은 불가능하다.[6]

아무튼, 시작하기 전에 간단한 MMD 신텍스는 몇 가지 알아두는 것이 좋다. 이게 *lightweight*인 이유는 알아두어야 할 것이 몇 되지 않는다는 것 때문이다.

일단 멀티마크다운 파일은 플레인 텍스트이다. 문단 첫 행은 1번 컬럼에 있어야 한다. 만약 행의 처음에 공란이 있으면 특별한 문단으로 간주할 것이므로 원하지 않는 결과가 나타날 수 있다. 또한 문단 구분은 하나의 빈 줄을 두어야 한다. 이것은 라텍의 경우와 같다. 일부 마크업에서 빈 줄을 중요하게 취급할 때가 있으므로 주의하자. 예를 들면 그림을 삽입할 때 빈 줄이 주어져야 플로트로 취급되며 enumerate도 빈 줄 다음부터 시작해야 `\begin{enumerate}`로 제대로

[5] https://rawgithub.com/fletcher/human-markdown-reference/master/index.html

[6] ConTEXt의 `\externalfigure`는 이게 되는데……

변환된다. 문단이 새로 시작하지 않으면 이렇게 되지 않을 수 있다. 눈으로는 보이지 않지만 공연한 스페이스를 넣어서 빈 줄이 빈 줄이 아닌 상황을 만드는 경우가 가끔 있다. 이럴 때는 메뉴의 Format → Convert → Strip Leading Tabs나 Multiple Spaces to Space 기능을 사용해서 소스를 깔끔하게 유지한다. 공백 글자를 시각적으로 확인하기 위해 Show Invisibles를 켜두는 것도 좋은 선택이다. 메뉴의 Format → Options → Show Invisibles를 활성화한다.

기호 입력에서 따옴표에 주의해야 한다는 것은 이미 언급하였다.

1. --와 ---가 문장 중에 사용되면 라텍에서와 같이 en-dash, em-dash이다.
2. ---가 문단으로 사용되면(즉 아래위로 빈 줄이 있으면) 가로선을 하나 넣는다.
3. 라텍에서 \'e와 같은 방법으로 입력하던 latin-1 확장 문자들은 문자 자체를 직접 입력하도록 한다. 메뉴의 Edit → Special Characters를 통해서 맥 오에스의 문자표에 접근할 수 있다. é. 이것은 입력기 메뉴의 "문자 보기 보기"로 문자표를 여는 것과 동일하다.[7]
4. 문단 첫 글자가 asterisk(*)일 때, 다음에 스페이스가 오면 itemize 리스트 항목으로 간주된다. 반면 글자가 바로 잇대어서 나오면 `\emph`로 변환된다. 요컨대, 스페이스가 중요하다.

이밖에 MMD 규약 일부를 소개하면 다음과 같다.

1. 장절 표제를 나타내기 위해 # 문자를 쓰는 것. 또는 github식으로 ==== 등을 다음 줄에 적어주는 방법도 있다. 가장 기본적인 mmd 문법이기는 하지만 스크리브너에서 장절 구성은 계층화된 폴더와 카드로 할 것이므로 그다지 신경쓰지 않아도 된다. 장절 구조화 (section 3.14) 절을 참조.
2. 글자의 강조. 별표를 쓰는 방법(*강조*, **볼드**)이나 언더스코어를 쓰는 방법(_강조_, __볼드__).
3. 코드. 라텍의 `\texttt`로 변환될 부분을 grave accent 문자로 둘러싼다.
4. 코드 블럭. 라텍의 verbatim 환경으로 변환될 부분은 행머리에 4스페이스를 두고 시작한다. 또는 grave accent 문자 세 개로 시작과 끝을 표시하는 방법도 있다.

[7]맥 오에스 매버릭스에서는 이전의 "문자 보기 보기"가 역시 이상하다고 생각했는지 "문자 보기 표시"로 바뀌었다.

5. 블럭 인용. 라텍의 quotation 환경으로 변환될 부분이다. > 문자를 행머리에 둔다.
6. 하이퍼레퍼런스. `[KTUG](http://ktug.org)`와 같이 입력하면 해당 위치로 링크와 각주가 붙는다. URL 자체만 입력하려면 `<http://www.ktug.org>`와 같은 방식으로.
7. 리스트 문단은 행 머리에 숫자가 오면 enumerate, 그 외의 문자(*, - 등)가 오면 itemize로 취급된다. 리스트 문단의 앞뒤로 빈 줄을 두는 것이 좋다.
8. 수학식은 행중 수식을 \\(와 \\) 사이에 둔다. 별행 수식은 \\[와 \\] 사이에 둔다. 수식에 쓰이는 마크업은 모두 라텍의 코드를 그대로 쓴다. 여러행 수식은 amsmath의 aligned나 split을 쓸 수 있다.

당장 알아야 할 것은 이 정도이다. 각주, 상호참조, 문헌목록, 그림, 표와 같은 더 확장된 기능은 해당하는 곳에서 설명하겠다.

2.4 설치

1. TEX을 설치한다.
   - MacTeX[8]을 다운로드받아 설치한 후 업데이트한다.
   - 업데이트 방법은 `TeX Live Utility` 앱을 실행한 후 메뉴에서 `Actions` → `Update All Packages`를 선택하고 관리자 패스워드를 입력한 후, (한참 동안) 기다리는 것이다. 반드시 업데이트하여야 한글 라텍 패키지들을 사용할 수 있다.
2. 멀티마크다운을 설치한다.
   - fletcherpenney.net[9]에서 Mac Installer와 Mac Support를 다운로드하면 `mpkg` 파일을 얻을 수 있다. 이를 실행하여 설치한다.
3. MMD LaTeX support 파일을 설치한다.

[8] http://tug.org/mactex

[9] http://fletcherpenney.net/multimarkdown/download/

- github/fletcher[10]에서 Clone in Desktop하거나 download ZIP을 선택하여 peg-multimarkdown-latex-support-master 폴더로 풀어놓는다. 이 폴더 전체를 ~/Library/texmf/tex/latex 아래 가져다둔다.
- ~/Library 폴더는 파인더에서 Shift-Cmd-G 단축키로 이동할 위치를 지정하는 방식으로 접근할 수 있다
- ~/Library 아래 texmf 폴더는 없으면 만든다. 그 폴더 아래 다시 tex 폴더를 만들고 다시 그 아래 latex 폴더를 만들어서 위의 다운로드받은 파일을 가져다두면 된다.

4. Scrivener

- App Store에서 구매하거나 제작사 웹사이트[11]에서 시험판 다운로드 후 설치.

한글 문서 작성을 위하여 필자가 제공하는 파일이 있다.

1. 20140211[12] 파일을 다운로드하여 압축을 해제하면 두 개의 폴더가 나온다.

a) 이 가운데 mmd-korean이라는 폴더를 ~/Library/texmf/tex/latex 아래 가져다둔다.

b) templates 폴더는 스크리브너 템플릿 파일들이다. 이 파일의 활용법에 대해서는 한글 mmd 템플릿 (chapter 5) 장에서 설명한다. 템플릿 활용하기 (section 3.22) 절을 참고하여 템플릿으로 등록해둔다.

2.5 스크리브너 설정

우리는 스크리브너를 텍스트 에디터로 사용할 것이기 때문에 여기에 맞게 몇 가지 설정을 행해주어야 한다. Scrivener의 Preferences를 열어서 다음 사항을 체크하자.

1. Corrections에서

- "Check spellings as you type in new projects"를 **check off**한다. (한국어 문서 작성을 주로 할 때)

[10]https://github.com/fletcher/peg-multimarkdown-latex-support

[11]http://literatureandlatte.com

[12]https://www.dropbox.com/s/f0t636c2r35lzj5/Scrivener-20140211.zip

- “Check grammar”나 “Correct spelling errors” 역시 한국어 문서 작성이 위주라면 체크하지 않는다.
- “Fix capitalization of sentences”를 **Check off**한다. (수학 문서를 주로 작성할 때)
- “Capitalize ‘i’ ” 역시 체크를 해제한다.
- “Use smart quotes”는 선택
- “Replace double hyphens with em-dashes”를 선택해제한다(**중요**).

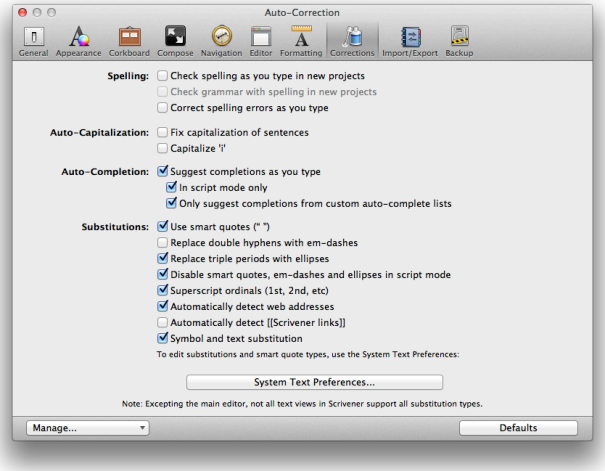

그림 2.4:

2. `Formatting`에서

- 편집창에서 자신에게 가장 편한 폰트와 여백을 설정한 후에
- “Use Formatting in Current Editor”를 누른다.

CHAPTER 3

스크리브너로 작업하기

3.1 화면 구성

스크리브너의 작업 화면 구성을 layout이라고 한다. 기본적으로 가운데 편집창(에디터), 왼쪽에 바인더, 오른쪽에 인스펙터가 있다.

- 바인더를 가리거나 열려면 툴바의 제일 왼쪽 아이콘을 클릭하거나, Option-Cmd-B 단축키를 이용하거나, 메뉴의 View-Layout-(Show/Hide) Binder를 선택한다.
- 인스펙터를 가리거나 열려면 Option-Cmd-I 단축키를 이용하거나, 메뉴의 View-Layout-(Show/Hide) Inspector를 선택한다.

그림 3.1: 툴바

툴바의 왼쪽에서 두 번째 아이콘은 "Show or hide collections"라는 것이다. 이것을 누르면 Binder/Search Result/Collections를 탐색할 수 있다.

세 번째 아이콘은 Show layout panel인데 레이아웃 패널을 열어서 현재 작업 화면의 상태를 등록해두었다가 (나중에 어떤 문서에라도) 적용할 수 있다.

작업 화면의 오른쪽 인스펙터 창은 여러 기능을 가지고 있다. 인스펙터 창 아래쪽에 있는 아이콘을 클릭하면 차례로

- Synopsis, General, Document Notes,
- Document References
- Keywords

- Custom Meta-data
- Snapshots
- Comments and Footnotes

등이 나타난다.

툴바의 여섯 번째 아이콘은 "full screen composition"이다. 화면 전체로 현재의 글쓰기 작업 창이 확대되어 오로지 글쓰기에만 집중할 수 있게 해준다. ESC로 나올 수 있다.

툴바의 열두 번째 아이콘부터 세 개는 작업 화면 보기 옵션이다. 각각 도큐멘트 보기, 코르크 보드 보기, 아웃라이너 보기인데, 하위 도큐멘트를 포함하고 있는 문서이거나 폴더인 경우 이 보기 옵션이 유용하다. 하위 도큐멘트가 없는 최말단 노드 도큐멘트에서는 기본 문서 보기만 된다.

3.2 편집창 설정

에디터의 테마, 폰트, 색상 등은 언뜻 보기에 지엽말단의 문제인 듯하지만 의외로 문서 작업의 생산성에 크게 영향을 끼친다.

스크리브너 설정 (section 2.5)에서 이미 언급하였지만 기본 글꼴을 잘 선택해 두는 것은 매우 중요하다.

한편, 다른 곳에서 붙여넣기를 하거나 외부 파일을 열었거나 할 때 폰트가 달라지는 경우가 있다. 이럴 때는 도큐멘트의 텍스트를 전체 선택한 후 메뉴에서 Document → Convert → Convert Formatting to Default Text Style을 선택해준다. 이것은 Format → Formatting → Apply Preset을 이용해서 할 수도 있다.

어떤 것이든 라텍 출력에는 영향을 미치지 않는다.

3.3 프로젝트, 폴더, 도큐멘트

하나의 문서에 여러 위계의 하위 문서가 포함될 수 있다.

1. 프로젝트(project): 최상위 문서 위계로서, 하나의 스크리브너 프로젝트 파일 (`.scriv`)로[1] 관리된다. 흔히 "스크리브너 파일"이라고 하면 프로젝트를 가리킨다. 문서 전체에 해당.

[1]엄밀히 말하면 파일이 아니라 폴더이지만 사용자 인터페이스로는 마치 파일처럼 다루어진다.

2. 폴더와 도큐멘트: 하나의 프로젝트는 여러 폴더나 도큐멘트를 포함할 수 있다.

    - 폴더와 도큐멘트의 차이는 본문을 포함할 수 있느냐 그렇지 않으냐 하는 것이다. 하위 도큐멘트만을 가지고 그 자체의 본문이 없는 것(시놉시스나 코멘트, 노트는 붙일 수 있다)을 폴더라고 하고 본문을 가지는 것을 도큐멘트라고 한다.
    - 도큐멘트는 다시 하위 도큐멘트를 포함할 수 있다. 그러므로 하위 도큐멘트를 가지는 도큐멘트와 그렇지 않은 도큐멘트로 나눌 수 있다.
    - 폴더는 스냅샷을 찍을 수 없다.

하위 도큐멘트를 가지는 도큐멘트에 본문이 없으면 폴더로 바꿀 수 있다. 바인더에서 도큐멘트 선택 → right click → Convert to Folder하면 된다. 같은 동작을 메뉴의 Documents → Convert → Convert to Folder로 선택할 수 있다.

폴더의 좋은 점은 기본 보기가 코르크 보드라는 것이다. 하위 문서들이 인덱스 카드 형태로 나열되므로 순서를 바꾸거나 하는 조작을 쉽게 할 수 있다. 코르크 보드 대신 아웃라인 보기 상태로 하여 내용을 확인하면서 비슷한 조작을 하거나 라벨을 붙이거나 하는 관리 활동을 할 수 있다.

"구조적인 글쓰기"는 별다른 것이 아니라 이렇게 계층적으로 구조화된 카드 목록으로 사고하고 글을 쓴다는 것이다. 라텍에서 강조하는 구조적 글쓰기의 핵심도 sectioning에 있다는 점을 생각해보라. 스크리브너는 Binder를 통해서 라텍보다 더 직관적으로 문서의 구조를 파악하고 더 쉽게 관리/유지/수정할 수 있게 한다.

바인더에 포커스를 둔 상태에서 현재 문서가 선택되어 있을 때, 여기서 Enter를 치면 같은 수준의 새로운 도큐멘트가 만들어진다.

하위 문서를 추가할 때는 현재 도큐멘트에서 right click하여 Add → New Text하는 방법이 있고(메뉴의 Project → New Text), 아니면 그냥 엔터쳐서 같은 수준의 문서를 만든 다음 원하는 도큐멘트 아래로 끌어다놓기 할 수도 있다. 이 끌어다놓기는 코르크 보드 상태에서 해도 되고 바인더에서 직접 조작해도 된다. 다만 바인더에서 직접 조작하는 것은 원하는 위치로 정확히 옮기는 데 약간의 노력이 필요할 경우도 있다. 도큐멘트 카드들을 합치고 섞고 하는 데는 역시 코르크 보드가 좋다고 본다.

이 폴더-도큐멘트 구조는 장절 구조화 (section 3.14) 절에서 다룰 주제와 연결되어 있다.

3.4 찾기와 바꾸기

스크리브너에서 찾기/바꾸기는 두 수준에서 이루어진다.

프로젝트 전체에 대해 찾기 Edit-Find-Project Search를 메뉴에서 선택하거나, 툴바의 Search Field를 이용한다.

해당 조건에 맞는 단어가 발견된 도큐멘트들이 Search Result에 나열된다. 이 상태에서 각 도큐멘트를 찾아가면서 적절한 조작을 행할 수 있다.

Edit-Find-Project Replace를 선택하여 프로젝트 전체에 대해 바꾸기를 할 수 있다. 이 때는 Regular Expression 바꾸기도 가능하다. 단, 이 조작은 취소할 수 없다.

참고로, Binder가 있던 위치에 Search Results가 나타난 다음 원하는 조작이 끝난 후 바인더를 다시 여는 방법은? 오른쪽 아래의 작은 x 표시를 누른다.

도큐멘트 내에서 찾기와 바꾸기 Cmd-F를 누르면 찾기/바꾸기 창이 열린다. 다음 찾기는 Cmd-G

RegEx 찾기 바꾸기를 하려면 찾기/바꾸기 창의 Find Option을 끌어내려보면 선택할 수 있다.

이밖에 '시놉시스에서 찾기' 라는 메뉴도 있다. 툴바의 검색 필드를 풀다운하면 찾을 대상(시놉시스, 타이틀, 메타데이터, 노트 등)을 선택할 수도 있다.

3.5 주석, 시놉시스, 노트

글을 쓰는 동안 주석(comment)을 붙여두는 것은 아주 중요하다. 왜 하필이면 이 단어를 선택했는지를 본문에서는 설명할 수 없으나 주석을 붙여둠으로써 이 메타정보를 이용하여 사고의 연속성을 확보하고 문맥과 논리를 검토할 수 있기 때문이다. 라텍에서 %로 시작하는 주석문을 활용할 것을 권유하는 이유이다.

스크리브너의 주석은 매우 다양한 종류가 있다.

시놉시스 가장 중요한 주석이다. 폴더와 도큐멘트에 붙일 수 있으며 해당 도큐멘트에 대한 포괄적 주석이 된다. 코르크 보드에서 카드 상태로 볼 때

시놉시스는 즉시 확인할 수 있다. 인스펙터 창에서 시놉시스를 확인하고 수정할 수 있다.

도큐멘트 노트 시놉시스에 간략한 주의와 의도가 들어간다면 이 영역에는 문서 전반에 관한 상세한 정보를 넣을 수 있다. 역시 인스펙터 창으로 확인하고 수정할 수 있다.

코멘트 특정 문장과 단어에 붙이는 주석이다. 툴바 가운데쯤 노란색 말풍선 모양의 아이콘으로 붙일 수 있고 역시 인스펙터 창에서 확인 가능하다. 특히 코멘트는 필요에 따라 각주로 전환할 수 있다. 단축키는 Shift-Cmd-*. 인스펙터 코멘트를 라텍 출력물에 포함시킬 것이냐 하는 문제는 컴파일 옵션으로 결정할 수 있다. 이 문서에서는 제외하도록 설정하였다.

Annotation 작성 중인 문장 가운데 넣는 주석이다. 필요하다면 코멘트로 변환할 수 있다. Shift-Cmd-A.

각주 Ctrl-Cmd–8 단축키로 각주를 삽입할 수 있다. 필요하다면 코멘트나 인라인 주석으로 변환가능하다. Shift-Cmd-F 단축키로 인라인 각주를 넣을 수도 있다.[2] 각주는 라텍으로 변환할 때 포함되는 것이 기본값이다. 각주를 코멘트로 변환할 수 있다.[3]

MMD는 자체 문법으로 각주를 지원하기도 하므로 그것을 그대로 사용해도 각주를 달 수 있다[4].

코멘트들 사이의 변환은 메뉴의 Format → Convert 아래를 찾아보라.

3.6 스크리브너의 컴파일 이해

스크리브너에서 입력된 텍스트는 일정한 규약에 따라 라텍으로 변환된다. 이 변환 작업을 "컴파일"이라고 부른다.

우리가 관심을 가지는 것은 한글 라텍 문서이다. 이를 위해 고려해야 할 요인이 몇 가지 된다.

---

[2]각주. 기본값은 인스펙터 각주이고 인라인 각주는 본문에 포함된다.

[3]인라인 각주를 이런 식으로 넣는 것도 가능하다.

[4]이 각주는 MMD 마크업으로 붙인 것이다.

1. Compile (메뉴의 File → Compile 또는 툴바의 컴파일 아이콘 또는 Option-Cmd-E) 창을 열고 Compile for를 `MultiMarkdown` → `LaTeX`으로 선택하고 All Options를 누른다.
2. Format As는 별도의 템플릿이나 설정을 이용하지 않았다면 Custom이 되어 있게 한다.

이 상태에서 compilation options를 설정하는 방법을 알아본다.

Contents 컴파일에 포함될 프로젝트의 문서를 고를 수 있다. 디폴트는 Draft 폴더 아래에 있는 모든 문서이다. 이 상태로 둔다.

Separators 디폴트 상태로 둔다.

Formatting Section Type에 세 종류가 있을텐데 이것은 각각 "폴더", "하위 문서가 있는 도큐멘트", "도큐멘트" 들이다. 각각에 대하여 Title을 포함하게 설정한다. 즉 Title을 모두 체크해준다. 이렇게 하면 도큐멘트의 타이틀이 라텍의 섹션 타이틀이 되게 할 수 있다. 이렇게 하는 것을 권장한다. 장절구조화 (section 3.14)에서 일부 도큐멘트에 대해서 타이틀을 제외하는 방법을 소개하겠다.

Layout, Transformation 특별한 것이 없으면 그대로 둔다.

Replacement 일괄 교체 지정이다. 여기에 지정된 것은 컴파일 전에 모두 일괄 변환된다. 이 기능을 활용하는 방법은 replacement의활용 (section 3.9)에서 다룬다.

Statistics 디폴트 그대로 둔다.

Footnotes & Comments inspector comments와 inline annotations를 출력에 포함할 것인지를 결정하여 체크한다.

Compatibility beamer 문서를 작성하는 경우가 아니면 XSLT post-processing을 체크하지 않는 쪽을 권장한다.

LaTeX Options 중요. 디폴트는 memoir로 되어 있을 것이다. 이 상태 그대로 컴파일하면 한글을 처리할 수 없다. 한글화 설정 (section 3.7) 절에서 상세히 다룰 것이므로 LaTeX Document class를 None/Use Metadata로 해두자.

Meta-Data 중요. None/Use Meta-data (subsection 3.7.2) 절에서 상세히 다룬다. 이 절에서 설명하는 내용을 참고하여 메타데이터를 잘 입력해야 한다.

3.6.1 다른 형식으로 내보내기

Compile for에서 선택할 수 있는 내보내기 방식은 다음과 같다.

1. `MultiMarkdown`: 플레인 텍스트 형식의 MultiMarkdown 파일이 만들어진다. 그림과 같은 부수 파일이 있으면 폴더를 만들고 그 안에 MultiMarkdown 소스 파일과 부수 파일이 저장된다. export된 MultiMarkdown 파일은 텍스트 에디터로 편집할 수 있는데 확장명이 붙어 있지 않으나 내보내기 옵션에서 `.txt`를 붙이게 할 수 있다. 대체로 말해서 `.txt`를 붙여두면 에디터로 열어보는 데 편하다. 이 파일에 대해서 후처리 조작은 다음과 같은 것이 가능하다.
    - `mmd2tex`: .tex 확장자가 붙는 라텍 파일로 변환한다.
    - `mmd2pdf`: 먼저 .tex으로 변환한 후에 latexmk를 실행하여 pdf를 만들어낸다.
    - `mmd`: HTML로 변환한다.
2. `MultiMarkdown` → `PDF`: 임시 폴더에 .tex을 생성하고 pdflatex을 실행하여 만들어진 PDF 파일만 저장한다. 매우 깔끔한 방법이기는 한데 한글 코드 문제 (section 6.8) 절에서 보듯이 "한글 제목의" 한글 파일을 처리하는 데 약간 문제를 일으킬 때가 있다.
3. `MultiMarkdown` → `LaTeX`: 이미 설명하였다. 생성되는 파일은 정상적인 라텍 파일이므로 이것을 TeXshop으로 열어서 편집하거나 컴파일하거나 할 수 있다. MultiMarkdown으로 내보내기 하여 `mmd2tex`한 결과와 같다.

이밖의 다른 형식으로 내보내기하는 것은 이 글의 관심사에 해당하지 않으므로 여기서 더 언급하지 않는다.

3.7 한글화 설정

스크리브너가 mmd-support를 통해서 기본적으로 제공하는 LaTeX 출력 옵션은 article, memoir, manuscript, custom 네 종류가 있다. 이 중 memoir가 디폴트이다. article, memoir, manuscript로는 한글을 쓸 수 없다.

한글 라텍 문서로 컴파일되게 하는 방법을 생각해보자.

3.7.1 custom document class를 이용한다

document class를 custom으로 선택하면 Header, Begin-document, Footer라는 세 입력 필드가 열린다. 이것을 이해하려면 MMD가 tex으로 변환하는 룰을 알아야 한다.

보통 MMD 파일의 Meta-data는 다음과 같은 모양을 하고 있다.

```
latex input: header
Title:       title text
Author:      author's name
latex input: begin-doc
latex footer:footer
```

멀티마크다운은 이 정보를 가지고 tex 파일로 변환하면서 다음과 같은 조작을 행한다. (footer는 `\end{document}` 직전에 삽입된다.)

```
\input{header}
\def\mytitle{title text}
\def\myauthor{author's name}
\input{begin-doc}
```

자, 이제 `header.tex`에는 무엇이 들어가야 할까? 적어도 `\documentclass` 문장이 있어야 할 것을 짐작할 수 있다. 그리고 `begin-doc.tex`에는 `\begin{document}` 문장이 포함되어야 하는 것이다. footer는 `\end{document}` 직전에 오는 것들, 예를 들면 `\printindex` 같은 명령을 포함하고 있으면 될 것이다. 간단한 문서에서는 이런 부분이 필요없으므로 footer를 따로 지정하지 않는 경우도 있다.

그러므로 Header 필드에는 예컨대 다음과 같은 내용이 들어가면 되겠다.

```
\documentclass[10pt,a4paper]{article}
\usepackage[hmargin=1in,vmargin=1in]{geometry}
\usepackage{kotex}
\ifx가가%
\setmainhangulfont{NanumMyeongjo}
\fi
\usepackage{graphicx}
\usepackage{amsmath,amssymb,amsthm}
\usepackage{cite}
\usepackage{tabulary}
```

이밖에 더 필요한 스타일 등이 있으면 써넣는다. article이 아니라 oblivoir로 하려 한다면 거기에 맞게 써넣으면 될 것이다.

그리고 Begin-document 필드에는 다음 내용으로 해볼 수 있다.

```
\begin{document}
\title{\mytitle}
\author{\myauthor}
\maketitle
\tableofcontents
```

그런 다음에 컴파일하여 시험해보라. 패키지가 부족하여 에러가 생기는 경우가 있다면 Header 필드를 충실하게 하여 문제를 해결하면 된다.

3.7.2 None/Use Meta-data

이제 이런 식으로 조성된 세 개의 파일을 각각

- `mmd-korean-header.tex`
- `mmd-korean-begin-doc.tex`
- `mmd-korean-footer.tex`

이라고 이름을 바꾸고 이것을 TEX이 이해할 수 있는 위치에 가져다 둔 다음 이것을 다른 문서에서도 이용할 수 있게 하자. 파일을 설치할 위치는 `~/Library/texmf/tex/latex` 아래 적당한 폴더이면 된다. 원한다면 `texmf-local` 아래라도 상관없기는 하다.

이 파일들이 설치되어 있다면 이제 스크리브너 컴파일의 LaTeX Option을 None/Use Meta-data로 바꾸고 Meta-data 항목에서 표 3.1과 같이 설정한다.

표 3.1: 메타데이터 설정

key	value
latex input	mmd-korean-header
Title	타이틀
Author	저자명
base header level	2
latex input	mmd-korean-begin-doc
latex footer	mmd-korean-footer
use xelatex	yes

메타데이터의 각 항목은 자명하므로 설명을 요하지 않으나 base header level 이라는 것은 조금 다르다. 이에 대해서는 최상위 레벨의 명령 (subsection 3.14.2) 절을 참고하라.

앞서 제시한 몇 가지 한글 문서 템플릿은 이런 방법으로 작성할 수 있게 한 것이다. 그러므로 None/Use Meta-data 상태인지 주의를 기울이고 Meta-data에서 latex input 지정자를 잘 써넣는 것 정도면 한글 라텍 문서의 생성에 큰 문제가 없다.

가장 쉬운 것은 템플릿을 활용하는 것이다. 템플릿활용하기 (section 3.22) 절에서 자세히 다루겠다.

3.8 Project Meta-Data

Compile의 메타데이터와 구분되는 또 하나의 메타데이터가 있다. 이것은 프로젝트 전체에 적용되는 것으로 Project → Meta-data Settings로 연다.

여기에는 Project title, Abbreviated title, Author' s Fullname 등이 기본값으로 설정되어 있다.

Compile의 메타데이터에서 Title에 스크리브너와 라텍이라고 되어 있었다면 이것은 그 값을 프로젝트 메타데이터에서 가져오라는 뜻이다. 그러므로 프로젝트 메타데이터를 설정해두고 이용하는 것이 좋다.

이밖에 사용자 정의 메타데이터를 관리할 수 있다. 라텍 클래스나 스타일에 따라 사용자 정의 메타데이터를 이용하는 것이 필요한 경우도 있으나 이 문제와 스타일시트를 통한 포스트프로세싱에 대한 것은, 이를 이용하면 MMD의 활용폭이 대단히 넓어지지만, 이 글의 범위에서 다루지 않기로 한다.

3.9 Replacement의 활용

MMD는 말 그대로 아주 간단한 마크업 언어이기 때문에 경우에 따라 (수식이 아니라도) 라텍 명령을 직접 입력해야 할 때가 있다. MMD는 이를 위한 한 가지 예외를 허용하고 있는데 그것은 "HTML 주석문"을 이용하는 방법이다.

```
<!--\LaTeX\ code-->
```

이렇게 마크업하면 이 주석문은 있는 그대로 라텍 파일에 전달된다.

이것은 MMD가 갖추지 못하였지만 라텍 문서 작성을 위해서 꼭 필요한 기능, 예컨대 정리류 환경을 식자한다든가 할 때 유용하다.

```
<!--
\begin{theorem}
…
\end{theorem}
-->
```

스크리브너의 컴파일 옵션 중에 "Replacement"라는 것이 있다. 이것은 일정한 문자열을 컴파일하기 전에 일괄 변환하도록 하는 것이다. 이것을 잘 이용하면 위와 같은 상황에 매우 효율적으로 대응할 수 있다.

한 가지 예를 들어보자.

```
Replace:  \#\|(.*?)\|
With:     \<\!--$1--\>
RegExp: check
```

이렇게 정의한 경우 소스에서 `#|\LaTeX|`이라고 입력한 것은 일단 `<!--\LaTeX-->`으로 변환된 다음 최종적으로 `\LaTeX`으로 텍 파일에 찍히게 된다. 남용하는 것이 좋다고 할 수 있을지는 모르겠으나 적절하게 활용하면 대단히 유용한 기능이다.

하나만 예를 더 들어둔다.

```
Replace:  \#\{(.*?)\{
With:   \<\!--\\begin\{$1\}--\>
RegExp: check

Replace:  \#\}(.*?)\}
With:    \<\!--\\end\{$1\}--\>
RegExp: check
```

이렇게 정의한 후에

```
#{theorem{
…
#}theorem}
```

이렇게 입력하면 `\begin{theorem}`과 `\end{theorem}`을 얻을 수 있다. 이와 비슷한 방식으로 Replacement를 정의해서 쓰고 있는 이 문서의 예를 참고할 수 있다.

굳이 RegExp를 사용하지 않더라도 Replacement의 쓸모는 많다.

```
Replace:  @@LaTeX@@
With:   <!--\LaTeX-->
RegExp: uncheck
```

이 상태에서 `@@LaTeX@@`은 라텍 소스의 `\LaTeX`으로 전달될 것이다.

단, 중복 정의가 가져올 혼란이나 일상적으로 사용하는 평범한 문자이거나 MMD 마크업의 예약 문자를 사용함으로써 의도하지 않은 교체를 하게될 위험이 있으므로 주의를 요한다. 특히 문자열의 치환이 "smart" 하지 않고 강제적으로 이루어지는 강력한 것임을 고려해야 한다.

3.10 Research 폴더의 활용

기본적으로 스크리브너의 바인더에는 Draft, Research, Trash라는 세 개의 폴더가 준비되어 있다. 사용자가 추가하는 것도 당연히 가능하지만 이 세 폴더는 항상 사용하는 것이 좋다.

Draft 작성 중인 문서의 폴더와 도큐멘트들이 온다. 기본적으로 본문을 구성하는 핵심적인 도큐멘트를 여기에 두고 작업한다.

Research 글과 관련된 자료들을 두는 곳이다. 관련 문서, 웹사이트, 그림 등을 이 폴더에 넣어두고 필요할 때 참조한다.

자료를 Research 폴더에 넣는 방법을 알아보자.

1. 외부 파일 끌어다 놓기. 예를 들어 바탕 화면에 있는 png 그림이 있다면 그냥 끌어다가 폴더에 집어넣으면 된다.
2. Research 폴더를 선택한 다음 right click한 후 Add를 선택하면 Existing Files와 Websites가 있다.
3. 프로젝트 내의 도큐멘트나 폴더는 right click하여 Move to로 선택하여 옮길 수 있다.
4. 다른 프로젝트의 도큐멘트나 폴더도 "끌어다놓기" 한다.

5. 스크래치 패드에서 바로 Research 폴더로 문서를 가져다놓을 수 있다. 스크래치패드 (section 3.11) 절에서 상세히.

6. File → Import 하여 멀티마크다운 파일을 가져올 수 있다.

본문을 작성하다가 Research 폴더를 열어서 자료를 확인하고자 할 때 현재 편집창의 포커스가 옮겨가는 것이 불편할 때가 있다.

1. Research 폴더를 선택하여 right click하면 Open in Other Editor가 있을 것이다. 이렇게 열면 편집창이 아래 위의 두 부분으로 쪼개져서 열린다. 편집창 오른쪽 위에 있는 작은 아이콘 toggle split을 이용하여 둘로 나눈 다음에 Open in Bottom Editor하는 방법도 있다.

2. 어쩌다 보니 그만 편집창을 split하지 않고 포커스를 Research로 옮겨버렸다고 하자. Cmd+[를 누르면 직전에 편집하던 위치로 돌아간다. (메뉴에서는 View → Editor → Backward in History).

물론 편집창 split으로 아래 위에 서로 다른 도큐멘트를 둘 수도 있기는 하다. 그러나 그런 목적이라면 quickreference (section 3.12)가 더 효용성이 높기 때문에 편집창을 아래위로 쪼개는 것은 대체로 그림이 들어 있는 폴더나 Research 폴더를 위하여 활용하는 것이 좋다.

참고로, 스크리브너에서는 컴퓨터 카메라로 찍은 사진이나 오디오 노트를 자료로서 붙일 수 있다. 이것들은 도큐멘트에는 직접 들어갈 수 없는 미디어이므로 Research 폴더에서 작성해야 하고 본문에서 링크를 걸 수는 있다. 다만 이 링크는 라텍 출력에 반영되지 않는다. Project → New Media File 아래를 찾아보라.

3.11 스크래치 패드

스크래치 패드는 Shift-Cmd-Enter하면 바로 열 수 있다. 메뉴에서는 Windows → Show ScratchPad이다.

스크래치 패드가 처음 실행될 때 적당한 폴더를 하나 알려달라고 한다. 스크래치 패드의 모든 자료를 그 폴더에 저장한다. 스크래치 패드는 말 그대로 자료를 수집하는 창이고 문서나 프로젝트와는 독립적인 자료가 된다. 이것을 문서의 자료로 활용하려면 스크래치 패드의 "Send to Project"를 이용한다. 현재 열려 있는 스크리브너 프로젝트의 원하는 폴더로 해당 노트를 복사할 수 있다.

스크래치 패드에는 note라는 형태로 자료들이 저장된다. [+] 버튼을 눌러서 노트를 만들고 여기에 필요한 걸 뭐든지 끌어다놓는 것이다.

리서치 폴더와 비슷하지 않나 할지 모르나, 결정적인 차이점은 다른 애플리케이션이 실행될 때도 스크래치 패드가 떠 있다는 것이다. 다른 앱을 실행하다가 작성중인 문서로 이걸 가지고 가고 싶을 때 스크리브너로 전환할 필요없이 스크래치 패드에 집어넣으면 된다. 상당히 강력한 자료 수집 툴이다.

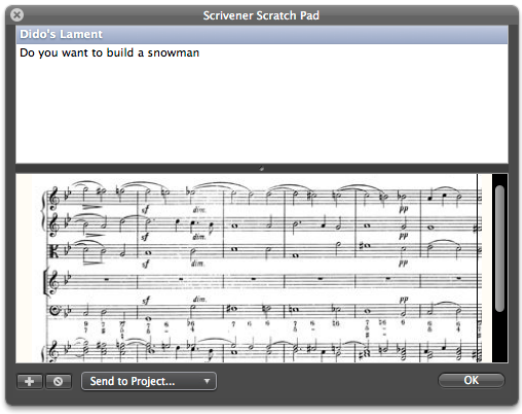

그림 3.2: 스크래치 패드

3.12 QuickReference

문서를 작성하다가 다른 도큐멘트에 기술된 내용, Research 폴더에 넣어둔 자료, 문헌목록만을 모아둔 도큐멘트를 잠시 참고하고 돌아와야 할 때가 있다. 현재 작업중인 편집창을 그대로 둔 채로 이런 "부수 문서"를 열어서 보는 창을 QuickReference라 한다. 해당 문서를 편집할 것이 목적이 아니라 참고할 것이 목적이다. (수정이 불가능하다는 의미는 아니다.)

QuickReference를 활용하는 두 가지 예를 들자면,

1. 번역문을 작성할 때. 원문을 QuickReference로 열어두고 작업 화면 옆에 띄워둔 채로 작업한다.
2. 문헌목록의 citation key를 확인하기 위해서 문헌목록 도큐멘트를 QuickReference로 연다.

예를 들어 '문헌목록' 이라는 이름의 도큐멘트가 있고 이것을 잠깐 열어서 내용을 참고하고 싶다고 하자. 해당 도큐멘트를 right click하여 Open → as QuickReference하면 된다. 이 때 편집창의 포커스가 잠시 문헌목록이라는 곳으로 옮겨졌으므로 Cmd-[로 이전 위치로 돌아간다. 이제 QuickReference에서 citation key를 찾으면서 글쓰기를 계속한다.

다른 프로젝트의 문서라도 QuickReference로 열 수 있다. Research를 위해서 가져다둔 PDF 문서도 QuickReference로 열린다. QuickReference 창을 여러 개 열어도 된다. 메뉴에서는 Document → Open → as QuickReference로 접근한다. 또는, 툴바의 여덟 번째 아이콘을 누르면 현재 편집창에 있는 문서를 바로 QuickReference로 열어준다.

3.13 Collections

스크리브너의 콜렉션이란 "특정한 속성을 지닌 도큐멘트들만 모아서" 관리하는 도구이다. 예를 들어 소설에서 특정 주인공 Alice 양이 있다고 하자. 이 주인공이 나오는 신만 모아서 Alice라는 콜렉션을 만들어두면 나중에 이 주인공의 행적을 추적하는 데 도움이 된다.

콜렉션에서 폴더는 의미가 없으므로 도큐멘트들만 들어가는데 특정 콜렉션에 모을 문서를 Cmd-click해서 right click → Add to Collection 하면 된다. 메뉴에서는 Documents → Add to Collection.

3.14 장절 구조화

스크리브너에서 장절 표제를 마크업하는 방법은 두 가지가 있다.

첫째는 스크리브너의 폴더/도큐멘트 트리 구조와 상관없이 MMD 마크업을 바로 이용하는 것이다.

```
# 1수준 헤더 #
## 2수준 헤더 ##
```

또는

```
1수준 헤더
=======
```

2수준 헤더

이렇게 마크업한다.

텍스트 본문(body)에 이러한 마크업이 들어가면 그것이 기록되는 위치가 어디든 상관없이 장절 명령으로 바뀐다.

두 번째 방법은 스크리브너의 폴더/도큐멘트 구조 자체를 장절 구분과 일치시키는 것이다. 각 도큐멘트나 폴더에 붙는 타이틀이 장절 표제로 변환된다.

첫 번째 방법은 문서의 구조가 복잡하면 나중에 수정하기도 어렵고 논리적이지도 못하다. 따라서 여기서는 두 번째 방법만을 사용하기로 한다.

3.14.1 헤더를 타이틀로 포함

스크리브너의 컴파일 이해 (section 3.6) 절에서 이미 언급한 컴파일 옵션 가운데 Formatting 항목에 주의한다. 여기 나오는 세 종류의 폴더 전부에 대하여 Title을 체크해주면 폴더와 도큐멘트의 타이틀이 장절 표제로 바뀐다.

그런데, 어떤 도큐멘트(카드)에 붙은 타이틀은 장절 표제로 변환되지 않아야 하는 경우가 있다. 대표적인 것이 문헌목록을 위해 만든 카드이다. 여기에 적힌 문헌 목록들은 MMD에 의하여 자동으로 `thebibliography` 환경으로 변환된다. 그러면 라텍에서 참고 문헌이라는 표제를 역시 자동으로 붙여준다. 이 카드의 타이틀을 또 섹션 명령으로 변환하면 라텍 파일에는

```
\section{문헌목록}
\begin{thebibliography}{00}
```

이렇게 찍히게 되고 그 결과 의미없는 절표제가 붙게되는 사태가 발생한다.

이를 피하려면 해당 카드에 대해서만 "Compile As-Is"를 활성화시켜 준다. 이것은 인스펙터 창의 Synopsis 아래에서 찾을 수 있다.

이 문서에서는 abstract를 쓰기 위하여 맨 처음 도큐멘트('개요')를 이와 같은 방식으로 포함하였다.

3.14.2 최상위 레벨의 명령

MMD의 논리로는 장절 표제는 header level일 따름이다. 예를 들어 제1수준 헤더로 마크업된 것을 라텍의 `\part`, `\chapter`, `\section` 가운데 어떤 명령으로 변환되도록 할 것인가?

이것을 지정하는 값이 Compile 옵션, Meta-data에 있는 `base header level`이라는 것이다. 이 값을 1로 하면 `\part`, `\chapter`, `\section` 순으로 각각 1, 2, 3 수준의 헤더가 변환된다. 이 값을 2로 하면 `\chapter`부터 시작한다. 즉, chapter가 있는 문서이면 2로 하고 `\section`부터 시작하는 문서이면 3으로 한다.

때에 따라 최상위 수준에 폴더가 있고 그 폴더 한 수준 아래 있는 도큐멘트들로 문서를 만들되, 스크리브너의 배치상 제2수준인 것을 라텍에서는 최상위 레벨 명령으로 들어오게 하고 싶은 경우가 있다. 이 때는 Formatting에서 최상위 레벨을 제외하고 제2수준부터 타이틀을 붙이게 설정하여야 한다. 그림 3.3은 이런 경우의 설정이다. 그리고 base header level을 원하는 값보다 1 적은 수로 지정한다. 예를 들어 최상위 명령이 `\section`이면 값을 2로 지정하는 것이다.

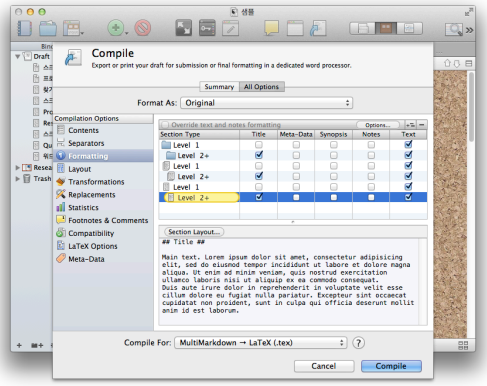

그림 3.3: 2수준 도큐멘트를 최상위 명령으로 내보내는 트릭

3.15 표제 목록으로 시작하기

가끔 장절 표제만 주르륵 나열하는 것으로 글쓰기를 시작하는 수가 있다. 글의 논리적 구조를 일단 대략 스케치하고 그로부터 각 절의 내용을 채워나가는 식으로 작업하는 것이다. 예를 들어 다음과 같은 글을 하나 구상하였다고 하자.

```
# 서론 #

# 본론 #
## 문제의 소재 ##

## 해결책 ##

# 결론 #
```

텍스트 에디터로 위의 내용을 적어서 abc.mmd라는 파일로 저장하였다고 하자. Draft 폴더에 포커스를 맞추고 메뉴의 File → Import → MultiMarkdown file 에서 이 파일을 찾아 불러오면 파일 이름을 폴더로 하여 각 표제별로 하나씩의 도큐멘트로 인덱스 카드를 만들어주는 것을 볼 수 있다. 이 상태에서 각 카드에 내용을 채우면 글이 완성된다.

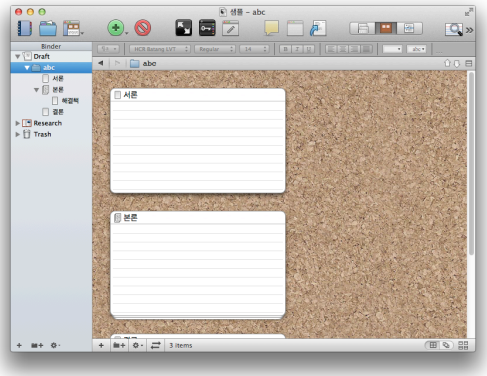

그림 3.4: mmd 파일 불러오기

이와 똑같은 것을 OPML 파일에 대해서도 할 수 있다. 이것이 좋은 것은 많은 마인드맵 앱들이 OPML로 저장할 수 있기 때문이다. 이것이 의미하는 것은 마인드맵으로 자신의 논문 구성(목차)을 스케치한 다음 이것을 글쓰기에 바로 적용할 수 있는 상태로 만들 수 있다는 것이다.

3.16 카드 합치기와 쪼개기

인덱스 카드 하나가 하나의 도큐멘트이다. 글을 쓰다보면 이 중 일부를 쪼개거나 합쳐야 할 때가 있다.

합칠 두 개의 도큐멘트를 선택(Cmd+click)하면 코르크 보드 보기 상태가 된다. 여기에서 메뉴의 Documents → Merge를 실행한다.

현재 커서가 있는 위치에서 새로운 도큐멘트로 쪼개려면 Documents → Split → at Selection하면 되는데 이것은 Cmd-K 단축키가 좋다.

3.17 각주

스크리브너에서 각주는 코멘트의 일종이다. 주석, 시놉시스, 노트 (section 3.5) 절에서 이미 다루었다. 그 나머지는 라텍이 처리해주는 것이 옳다.

예컨대 MMD에 미주(endnote) 기능이 있다고 하더라도 이것을 사용하지 않고 라텍 스타일 파일이 스스로 `\usepackage{endnote}`로 처리하게 하는 것이 바람직하다.

3.18 그림

라텍에서는 그림을 어떻게 넣었는지 생각해보자.

1. 그림은 tikz같은 드로잉 언어를 이용하여 그릴 수도 있다. `png`, `jpg` 같은 그림을 삽입할 수도 있다.
2. 그림을 떠다니게 할 수도 있고 그렇지 않을 수도 있다.
3. 캡션이 붙는 경우도 있고 그렇지 않은 경우도 있다.
4. 나중에 그림을 cross reference로 참조하는 경우도 있다.
5. 그림이 문단의 일부로 포함되어 배치되는 경우도 있다.
6. 배경그림이나 로고와 같은 장식적 요소로 쓰이기도 한다.

이외에도 그림을 다루는 데 수십 가지 예상가능한 문제들이 존재한다.

이제, MMD로 문서를 작성하려 한다면 이 문제에 대해서 사태를 최대한 간단하게 만들어야 한다. 이를테면,

1. tikz같은 드로잉 언어를 써야 한다면 그것은 MMD의 일이 아니다. 그림에 해당하는 부분만 오리지널하게 LaTeX 파일로 만들어서 pdf를 제작한 다음 그것을 보통 그림처럼 불러오는 방식으로만 쓰자.
2. 배경그림이나 장식적 로고 등은 MMD의 일이 아니라고 생각한다. 그런 것은 라텍 스타일 파일이 알아서 해주어야 할 문제이다.
3. 일단 모든 그림을 떠다니는 그림으로만 넣는다. 캡션은 붙일 수도 있고 안 붙일 수도 있지만 원칙적으로 다 붙이는 것으로 하자. wrapfigure 같은 "글쓰기"와 관계없는 페이지 디자인의 문제 역시 MMD의 소관사항이라고 보지 않는다.
4. 단, "글자 대신 쓰인 그림"은 그대로 그림을 글자 사이에 박아넣는다.

이 정도의 결심이 확고해야 한다. 왜냐하면 이 이외의 것을 더 하기 시작하면 그것은 절대로 "글쓰기"가 담당할 영역이 아닐 것이기 때문이다.

그림을 준비하는 것 역시 별개의 문제이다. 화면 스크린을 찍든, 사진을 복사해오든, 인터넷에서 다운로드받든, GeoGebra나 asymptote로 그림을 그리든 어떤 식으로든 원하는 그림을 준비하는 것은 확실히 저자의 일이지만 그것은 글쓰기의 준비과정이지 글쓰기 자체는 아니다. 이렇게 준비된 그림들은 전부 Research 폴더에 넣어둔다.

그림 이름을 지을 때 주의할 것이 있다. 라텍은 그림 이름에 공백이나 아스키 문자 아닌 것이 오면 좀 싫어한다. 아스키 문자만으로 공백없이 그림 이름을 붙여두어야 나중에 편하다.

가끔 그림이 `png`, `jpg`, `eps`, `pdf`가 아닌 경우가 있다. 예를 들면 `.tiff` 같은 것. 그런 확장자를 가진 그림을 사용해야 하는 상황이면 ImageMagick의 convert 같은 툴을 이용해서 비트맵 그림은 `png`로, 벡터 그림은 `pdf`로 변환해두도록 한다. 역시 작업의 편의를 위하여 그림 파일 포맷을 이 두 가지만 쓰는 것이 좋다. 혹시 `jpg` 사진이 있다면 미리 `png`로 변환해두는 쪽이 좋다. 그리고 Research 폴더에는 변환된 사진만 유지한다. Research 폴더에 드래그앤드랍하더라도 원본 그림이 사라지는 것이 아니고 원본을 지우더라도 복사본은 스크리브너 프로젝트 안에 남아 있다.

이제 그림이 필요한 곳이 나오면 미리 한 행을 띄우고 Research 폴더에서 그림을 현재 위치로 끌어다 놓는다. 이 작업은 (당연하지만) 편집창을 아래위로 나누어둔 상태에서 하는 것이 가장 좋다.

그림이 들어가면 크기를 조절하자. MMD에서는 그림 크기를 픽셀 단위로만 줄 수 있고 `0.5\textwidth` 같은 방식으로는 줄 수 없다. 그림에서 right click 하면 Scale Image를 할 수 있다. 대략 가로 크기 픽셀을 정해주면 나머지는 알아서 한다.

그림이 놓인 위치를 잠깐 검토한다.

1. 그림 앞에 빈 줄 하나가 있어야 한다. 그래야 figure 환경 안에 들어간다.
2. 그림 앞에 공백 스페이스가 없어야 한다.

이제 그림이 놓인 다음 줄에 빈 줄 없이 캡션을 `[캡션]`과 같은 방법으로 붙인다. 그리고 빈 줄 한 줄을 두고 본문을 계속한다.

MMD의 자동 라벨링은 그림 이름을 라벨로 사용한다. 예를 들어 test.png라는 그림에 대해서 `\label{test.png}`가 붙는다. 표와 그림의 참조 (subsection 3.20.3) 절 43 페이지에서 이것을 참조하는 방법을 설명한다.

한편, 글자 대신 쓰인 그림이라면 그냥 그림을 끌어다놓고 크기 조절을 해주는 것으로 충분하다. 이것은 캡션이나 라벨 등이 전혀 붙지 않고 그냥 그림만 들어가게 된다.

사실 MMD 자체는 그림을 삽입하는 몇 가지 방법이 더 있다. 그러나 거기에 대해서 말하지 않으려 하는데 문제가 복잡해지는 것을 바라지 않기도 하거니와 라텍의 관점에서 불필요한 것도 많기 때문이다.

문단의 일부로 파고드는 그림 (wrapfigure) 같은 것은 MMD의 범위를 넘어선다. 만약 그것을 꼭 해야 한다면 LaTeX Raw Coding으로 처리하여야 할 것이지만, 원고를 작성하는 시점에서 이런 것으로 고민하지 않기를 바란다. 그냥 끌어다놓기만으로 작업하자.

3.19 표

표 (table)는 라텍에서도 어려운 주제이다.

대략 세 종류의 표를 생각해보자.

1. 매우 긴 표이지만 주로 데이터의 나열로 되어 있는 것.
2. 비교나 대조를 위하여 작성된 표.
3. 긴 텍스트가 들어가고 셀의 머지가 많은 복잡한 표.

라텍에서도 그렇지만 세 번째 유형의 표는 그리는 것이 쉽지 않다. 그런 유형의 표를 전문적으로 작성해주는 툴을 이용하여 그림으로 가져오는 편이 안전하다.

MMD가 표 문법을 제공한다. 간단한 표를 바로 그릴 수 있다. 별도의 XSLT를 사용하지 않을 경우, 즉 기본 memoir.xslt인 경우 MMD 표는 tabulary 패키지와 booktabs를 이용하는 표로 변환된다. 세로줄이 없고 첫 줄이 조금 굵게 표시된다. 코딩은 다음과 같이 하고

```
|              |          Grouping           ||
First Header   | Second Header | Third Header |
 ------------- | :-----------: | -----------: |
Content        |          *Long Cell*        ||
Content        |   **Cell**    |         Cell |

New section    |     More      |         Data |
And more       |            And more         ||
[Prototype table]
```

표 3.2: Prototype table

	Grouping	
First Header	Second Header	Third Header
Content	*Long Cell*	
Content	**Cell**	Cell
New section	More	Data
And more	And more	

표 3.2가 실제 그려지는 표이다.

이 절에서 지적하고자 하는 것은, 스크리브너 메뉴의 Format → Table로 그리는 표는 MMD가 아니라는 것이다. 즉, 그렇게 해서 만들어진 표는 라텍으로 변환되지 않는다.

표에 한하여, LaTeX 코드를 그대로 쓰는 방법도 나쁘지 않다고 생각한다. 표 3.3을 보라.

이 방식의 장점 중 하나는 엑셀이나 오픈오피스 스프레드시트 데이터를 표로 가져오려 할 때 csv2latex이라는 유틸리티의 도움을 받아 copy&paste할 수 있다는

표 3.3: 시험삼아 만든 표

이름	나이	성별
홍 길 동	18	남
성 춘 향	15	여
임 꺽 정	35	남
허 초 희	20	여

점이다. 블로그 글[5]을 참고하라.

3.20 라벨과 상호참조

MMD가 라벨과 상호참조 기능을 일부 제공한다.

3.20.1 장절 표제의 참조

장절 표제에 라벨은 자동으로 붙는다.

```
# 장절 표제 #
```

이렇게 마크업한 텍스트는 라텍에서

```
\section{장절 표제}\label{장절표제}
```

이와 같이 구현된다. 즉 장절 표제 텍스트에서 띄어쓰기를 다 제외하고 라틴 문자는 모두 소문자로 바꾸고 특수문자도 제외한 다음 자동으로 라벨을 붙인다.

강제 라벨링을 위해서는

```
# 장절 표제 [sec:mylabel] #
```

이렇게 코딩해야 한다. 스크리브너 도큐멘트 타이틀을 장절 표제로 사용하는 경우 표제 뒤에 `[sec:label]`을 추가해주면 이와 동일하게 동작한다.

장절 표제를 참조하려면 `[표제레이블][]`와 같이 코딩한다. 그러면 장절표제의참조 (subsection 3.20.1)와 같이 구현된다. MMD의 이 방식이 마음에 들지 않으면 LaTeX Raw Code로 입력하자. `<!--{\ref{장절표제}}-->`

`\pageref`이나 `\titleref`은 Raw Code를 사용하는 것이 좋다. 장절 표제의 참조.

[5] http://doeun.blogspot.kr/2013/12/csv2latex.html

3.20.2 수식의 라벨과 참조

MMD에서 수식은 \\(, \\), \\[, \\]만을 쓴다고 가정하자. 이 가운데 행중수식을 참고하는 경우는 없다고 보아도 무방하므로 별행수식의 경우를 생각한다.

단행의 별행 수식을 상호참조하려면 수식 번호를 붙여야 한다. 라텍에서 equation 환경을 쓰라고 하는 대목이다. 그러므로 Raw Coding에 가깝게 다음과 같이 입력하면,

```
<!--
\begin{equation}
a^2+b^2=c^2 \label{eq:test}
\end{equation}
-->
```

$$a^2 + b^2 = c^2 \tag{3.1}$$

식 (3.1)과 같은 결과를 얻을 수 있다. 참조를 위한 명령은 `\eqref`을 직접 써넣는 것으로 한다.

그러나 뭔가 일관성이 없어 보여서 `\autotag`라는 명령을 하나 정의했다. displaymath 환경이라도 수식 번호를 붙이게 하는 것이다. 이 명령은 `mmd-korean`에 정의되어 있다.

```
\\[
\text{Life} = \int_{\text{Birth}}^{\text{Death}}
\text{Choice}(t)\,\mathrm{d}t \autotag
\label{eq:test2}
\\]
```

$$\text{Life} = \int_{\text{Birth}}^{\text{Death}} \text{Choice}(t)\,\mathrm{d}t \tag{3.2}$$

그 결과는 식 (3.2)와 같다.

여러 줄 수식의 경우는 더 간단해보인다.

$$\begin{aligned} A &= b + c \\ &= F + X \end{aligned} \tag{3.3}$$

3.20.3 표와 그림의 참조

표 (section 3.19)에서 보인 대로, MMD 표는 마지막 줄의 [] 안에 오는 텍스트를 label로 한다.

그림의 경우 아래 위로 빈 줄이 들어간 플로트 그림은 그림 아래 []에 써넣은 표제가 캡션이 되고 그림 파일의 이름이 label이 된다. 그림 이름이란 Research 폴더에 있는 그대로의 이름을 normalize한 것으로 대문자는 모두 소문자로 바뀌고 공백은 없어지며 특수문자가 제거된 그림이름을 가리킨다. 따라서 Research 폴더에서 그림 이름을 잘 지어두는 것이 좋다. 공백 문자 없는 아스키 문자만으로 그림 이름을 지어야 라텍이 처리하기 쉽다. 그러므로 예컨대 그림 7.1에서 label은 `doyouwannabuildasnowman.png`이다.

참조 명령은 LaTeX Raw Code로 `\ref`를 쓰는 것이 좋다. 이 문서에서는 그림에 대해서 `\fref`, 표에 대해서 `\tref`을 쓰고 있다.

표 3.4: 자주 쓰이는 참조 명령

참조명령	비고
\ref	number
\pageref	no. of page
\eqref	(number)
\Sref	§ number
\Cref	chap. number
\tref	tablename. number
\fref	figurename. number
\titleref	title text

MMD가 cross reference 기능을 제공함에도 불구하고 `\ref` 명령을 그냥 쓰는 편을 권장하는 이유는 이 도구를 쓰는 이유가 복잡해지는 것이 싫어서이기 때문이다. 경우마다 조금씩 다르게 동작하는 참조 명령을 새로 익힐 필요가 없다. 특히 우리가 의도하는 최종출력이 HTML이 아니기 때문에 더 그러하다.

3.20.4 자동조사

이것은 MMD에서 제공하지 않으며 제공할 수도 없는 기능이다. 따라서 LaTeX Raw Code로 입력한다. \은, \는, 등…….

3.21 문헌목록의 활용

MMD의 bibliography 지원 기능을 요약하면 다음과 같다.

```
[p. 10][#fake]

[#fake]: Nobody, *Unseen Book*, Noname Press, 2014.
```

MMD 프로세서가 이것을 모두 처리해준다. 그러므로, 예를 들면 "문헌목록"이라는 이름의 도큐멘트를 하나 만들고(이것의 타이틀이 섹션으로 포함되지 않도록 Compile As-Is 옵션을 활성화한다), 여기에 해당 문서에서 사용할 문헌목록을 작성한다.

본문에서는 문헌목록의 참조자(citation key)를 이용하여 인용한다.

이렇게 만들어진 문헌목록은 다른 비슷한 글을 쓸 때 재활용할 수 있다. 문헌목록에 아무리 많은 문헌이 기록되어 있어도 MMD는 그 중에서 본문에 인용된 것만 추려서 **인용 출현순으로** 정렬하여 참고문헌 목록을 만들어준다.

만약 **인용 출현 순(order of citation)**이 아니라 알파벳순으로 정렬하려면 어떻게 해야 하는가? 그것은 MMD가 관여할 일이 아니라 라텍과 부수 유틸리티를 활용해서 해결할 문제이다. thebibliography 환경 안의 `\bibitem` 들을 알파벳순으로 정렬해주는 유틸리티 LaTeX-BibitemStyler[6]같은 것을 이용하는 것이 한 방법인데 이 역시 MMD와는 별 상관이 없는 문제이다.

BibTEX을 이용하는 방법도 고려해볼 수 있다. 알파벳순 정렬뿐 아니라 고급의 문헌 목록을 작성하는 데 매우 유용하고, 스크리브너로도 **조금만 유념하면** 이 유틸리티를 사용하도록 문서를 세팅하는 것이 가능하다. (단, 한글 문헌을 처리하는 데는 아직 적당한 해결책이 없다.) BibDesk라는 훌륭한 유틸리티를 MMD 작업에 맞게 설정하여 문헌 인용에 사용하는 방법도 있다. 이에 관한 사항을 더 자세히 다루는 것은 다른 글을 하나 더 쓰거나 하고 여기서는 이 정도로 줄이겠다. 특히 한글 문헌 목록을 제대로 구현하는 문제가 먼저 해결되어야 하는 것이기 때문이다.

문헌목록은 최종 단계에서 약간의 수작업을 요청할 가능성이 있다. 그것은 한글 라텍의 경우도 마찬가지다.

[6] https://code.google.com/p/latex-bibitemstyler/

3.22 템플릿 활용하기

스크리브너의 템플릿은 매우 쉽고 강력하다. Preferences로 설정하는 스크리브너 전역 설정 사항을 제외하고 한 프로젝트에서 설정한 모든 설정이 그대로 유지된다. 심지어 Replacement 설정 같은 것도 그대로 유지된다.

템플릿으로 등록하는 방법은 간단하다. File → Save As Template. 그 후로는 File → New Project에서 새로운 프로젝트 문서를 생성할 때 템플릿을 선택하여, 똑같은 설정을 가진 프로젝트 문서를 만들어낸다. 마치 스크리브너의 처음 설정이 무척 복잡한 것처럼 보이지만 단 한 번만 제대로 문서를 설정해두면 그 후로는 반복작업도 필요없고 간단히 글쓰기를 즉시 시작할 수 있다. 다른 사람이 작성한 잘 만들어진 템플릿을 등록하여 쓴다면 자신이 한글 설정 등을 고민할 필요조차 없다.

강좌를 진행하면서 모두 네 종류의 한글 문서 템플릿을 제작하여 제공하였다. 이에 대해서는 한글 mmd 템플릿 (chapter 5) 절에서 설명한다.

CHAPTER 4

마크다운 확장

4.1 리스트 문단의 마크업

리스트 문단의 마크업을 테스트해보자.

번호붙은 리스트

1. 아이템 첫 번째
2. 아이템 두 번째
    a) 서브아이템 첫 번째
    b) 서브아이템 두 번째
3. 아이템 세 번째
4. 아이템 네 번째

리스트가 일단 시작하면 새 행의 첫 컬럼에 숫자나 불릿 표지(*, -)가 오지 않을 때까지 리스트로 간주한다.

그래서 이런 문제가 생긴다. 예를 들면

```
\begin{enumerate}
\item
\item
\end{enumerate}
\begin{itemize}
\item
\end{itemize}
```

이런 식으로 서로 다른 리스트가 연이어 오게 하고 싶을 때 어떻게 하는가? MMD는 맨 첫 번째 리스트가 시작할 때 표지가 숫자인가 아닌가만 문제삼을 뿐

그 뒤에는 뭐가 오든 모두 똑같은 리스트라고 간주하기 때문에 위와 같은 문단 형태를 만들려면 두 리스트 사이에 무의미한 텍스트를 넣어주어야 한다.

```
1. 아이템
1. 아이템

<!--{}-->

* 아이템
* 아이템
```

1. 아이템
2. 아이템

- 아이템
- 아이템

이것이 아마 현실적인 해결책일 것이다.

한편, 스크리브너의 위지위그 편집 기능으로 작성된 리스트는 라텍에서 아무 의미가 없다.

4.1.1 항목의 나열

라텍의 description 환경과 비슷한 것은 다음과 같이 입력할 수 있다. 이 방법에서 중요한 것은 아이템들 사이에 빈 줄을 두는 것이다.

마크다운 John Gruber가 만든 경량화 마크업 언어.

멀티마크다운 Fletcher Penney에 의한 마크다운의 확장.

4.1.2 enumerate 패키지

마지막으로, oblivoir에서 즐겨 쓰던 `\begin{enumerate}[(1)]`와 같은 코딩은 어떻게 해야 하는가? 간단히 포기하고 안 쓰는 게 답이긴 하지만… Raw Coding으로 처리하는 방법이 있기는 하다. MMD스럽게 이 문제를 해결하는 방법은 이것을 지원하는 스타일시트(XSLT)를 만드는 것인데 아직 그런 스타일시트가 제작되지는 않은 것으로 안다.

4.2 인용문 환경의 마크업

인용문은 quote, quotation 환경에 대응된다.

```
> 장 폴 사르트르가 말하기를 “인생이란 출생과 죽음 사이의 끝없는
> 선택”이라고 하였다.
```

> 장 폴 사르트르가 말하기를 “인생이란 출생과 죽음 사이의 끝없는 선택”이라고 하였다.

flushright 같은 환경은 되도록 쓰지 않는 것이 좋다. 그러나 굳이 써야 한다면 Raw Coding하는 방법이 있다.

```
<!--\begin{flushright}-->
…
<!--\end{flushright}-->
```

운문(verse) 환경 역시 꼭 필요하다면 Raw Coding한다. 그런데 이에 대해 재미있는 기고가 있어 여기 링크를 남겨둔다. 즉 verbatim으로 묶을 부분을 verse로 대체하여 운문 환경을 마크업하는 것. 이 글[1]을 보라.

[1] http://doeun.blogspot.kr/2014/02/mmd-verse.html

CHAPTER 5

한글 mmd 템플릿

5.1 한글 문서 템플릿 네 개

필자가 제공하는 스타일 묶음[1]에는 다음과 같은 것들이 포함되어 있다.

1. mmd-korean
   - `mmd-korean-header.tex`, `mmd-korean-begin-doc.tex`
   - mmd-korean 템플릿.[2] (chapter있는 문서)
2. mmd-oblivoir
   - `mmd-oblivoir-header.tex`, `mmd-oblivoir-header.tex`
   - mmd-oblivoir 템플릿. (chapter 없는 문서)
3. mynovel
   - `mynovel.cls`, `mynovel-header.tex`, `mynovel-preamble.tex`
   - mynovel 템플릿.
4. beamer-korean
   - `mmd-beamer-korean-header.tex`, `mmd-beamer-korean-begin-doc.tex`, `mmd-beamer-korean-footer.tex`
   - beamer-korean 템플릿.
5. progress-thm.tex

[1] https://www.dropbox.com/s/f0t636c2r35lzj5/Scrivener-20140211.zip

[2] 장절표제와 페이지 스타일은 Progress님이 디자인하신 것으로 2013 KTUG 공주대 워크숍에서 발표된 것을 바탕으로 하고 있다.

- KTUG의 2013 공주대 문서작성워크숍에서 Progress님이 발표하신 "정리류 환경".
- mmd-korean과 mmd-oblivoir에서 이 정리류 환경 정의를 사용함.

5.2 소설 쓰기

소설 작가들은 어떤 도구를 쓸까? 알려진 바에 따르면 아래아한글 사용률이 압도적이라고 한다. 그런데 아래아한글 같은 도구를 이용해서 글을 쓰면서 예컨대 판타지 소설이라면 세계관의 관리는 어떤 식으로 하는 걸까? 별도의 파일을 만들어서 그때그때 참조하거나 보충하면서 작업하나?

아래아한글 같은 워드 프로세서는 사실 수많은 생각, 설정, 등장인물을 관리해야 하는 소설과 같은 문장을 쓰는 데는 독이다. 본문밖에 보이는 것이 없기 때문에 자신이 지난 번에 특정 인물을 죽였는지 살렸는지 감감할 때 이것을 효과적으로 통제할 수단을 이런 종류의 워드 프로세서는 제공하지 않는다. 혹시 우리나라 소위 "양판소"에서 설정붕괴가 유행이라면 작가가 사용하는 도구를 의심해봐야 하는 거 아닌가?

스크리브너의 강력한 주석 기능, 시놉시스 관리 기능, 버전콘트롤 기능, 그리고 폴더 관리 기능을 이용하면 소설을 좀더 짜임새있게 쓸 수 있지 않을까 기대한다. 사실 소설이란 작가의 머리 속에 있는 거대한 세계의 편린을 내보이는 것일 터이니, 그 "거대한 세계"를 온갖 자료들로 구축하고 그 중의 극히 일부를 소설이라는 형식으로 내보이는 부분에 Draft의 문장들을 배치하면 되는 거라고 본다.

mynovel.cls는 소설의 최종 출력물을 얻게 하는 라텍 클래스이다. 기본적으로 요즘 출판되는 소위 "장르소설" 출판물과 판면이 비슷하다. 스크리브너를 이용해서 작성한 소설을 mynovel.cls는 간단히 출판물 품위의 pdf 파일로 변환해준다.

5.3 수학식이 있는 문서 쓰기

수학 문서는 MMD와 스크리브너로 훌륭하게 작성할 수 있다.

행중 수식 \\(와 \\) 사이에 라텍 수식 매크로를 넣는다.

단행 별행 수식 \\[와 \\]로 displaymath를 넣는다.

수식 번호를 넣으려면 \autotag라는 mmd-korean 스타일에 정의된 매크로를 쓴다. 즉 equation 환경을 쓰지 않아도 된다.

여러행 수식 Raw Coding으로 amsmath의 여러행 수식 환경을 써도 좋다.

amsmath에서 수식 환경 내에서의 여러행 수식을 지원하는 alternative 환경들(aligned, gathered 등)을 써도 좋다.

수식 번호는 `\autotag`이 가능하면 그것을 쓴다.

정리류 환경은 모두 *환경*이므로, Replacement와 Raw Coding을 결합하여 사용하면 편하다. proof 같은 것도 결국 환경이다.

정리류 문단에 대한 정의는 라텍 스타일에서 별도로 해두어야 한다. 이 문서의 스타일인 mmd-korean에서는 `thm` 환경과 `proof` 환경을 정의해두고 있다.

정리 5.3.1 (피타고라스의 정리) 임의의 직각삼각형에서 빗변을 한 변으로 하는 정사각형의 넓이는 다른 두 변을 각각 한 변으로 하는 정사각형의 넓이의 합과 같다. 즉 $\angle C$가 직각일 때,

$$a^2 + b^2 = c^2$$

증명 이 사이트에 가면 100여개에 이르는 증명[3]을 볼 수 있다. 《주비산경》의 증명은 No.4[4]이다. ■

한 가지 팁이라 할 만한 것은 사용자 정의 명령을 활용하는 것이다. 프로젝트의 맨 처음에 MyDefinitions 정도의 이름으로 도큐멘트를 하나 만들고 이것이 섹셔닝하지 않도록 Compile As-Is를 선택한다.

그리고 이 도큐멘트 안에 Raw Code 형식으로 사용자 정의 명령을 넣어둔다.

```
<!--
\newcommand\ZZZ{\ensuremath{\mathbb{Z}}}
-->
```

이 도큐멘트에는 다른 것을 일절 넣지 않는다. 그리고 수식 작성 시에 이 명령을 활용한다.

$$x \in \mathbb{Z}, \forall x p(x)$$

수식 작성 자체는 라텍 수식 작성과 다를 것이 없다. 문서 [4]를 참고하라.

[3] http://www.cut-the-knot.org/pythagoras/index.shtml

[4] http://www.cut-the-knot.org/pythagoras/index.shtml#4

CHAPTER 6

스크리브너 활용

6.1 Marked앱을 이용한 미리보기

Marked 2[1]는 (Multi)Markdown Previewer이다. 당연히 라텍을 경유하여 컴파일한 PDF를 보여준다든가 하는 그런 Previewer가 아니라 마크다운 마크업을 실시간으로 HTML 콘텐츠로 보여준다는 것이다.

이 앱은 자체적으로 스크리브너 파일을 지원한다. 스크리브너 프로젝트 파일을 Marked 2로 열면 preview가 가능하다.

Marked 2에 대해서 언급하는 이유는 이 앱이 MathJax[2]를 이용하여 수식 미리보기가 가능하다는 점 때문이다. 생각보다 수식의 입력이 간단하지는 않다. 실수도 잦고 작성한 수식이 제대로 제 모양으로 나오는지 확인하는 것도 쉽지 않다. 수식이 많은 문서를 작성하는 라텍 저자가 컴파일을 자주 하는 이유가 수식에서의 오류 때문이다.

Marked 2의 Preferences의 Style에서 Include MathJax를 체크하고 설정을 저장하여야 한다.

스크리브너 파일을 Marked 2로 끌어가는 방법으로 연 다음에 Cmd-E를 누르면 스크리브너로 해당 파일을 열 수 있다. Preference의 Behavior에서 도큐멘트 타이틀을 미리보기의 헤더로 표시하도록 선택할 수 있다. 스크리브너에서 파일을 수정하면 마지막 수정된 부분으로 미리보기의 포커스가 자동으로 이동한다.

다만, Marked 2의 미리보기는 이를테면 *의미상* 동일한 모양을 보여줄 뿐이고 최종 출력물의 모양이 아니다. 이 점을 감안하고 사용하여야 한다. 이 글이 권유하는 바는 **수식 미리보기** 때문 그 이상도 이하도 아니다.

---

[1] http://marked2app.com

[2] http://mathjax.org

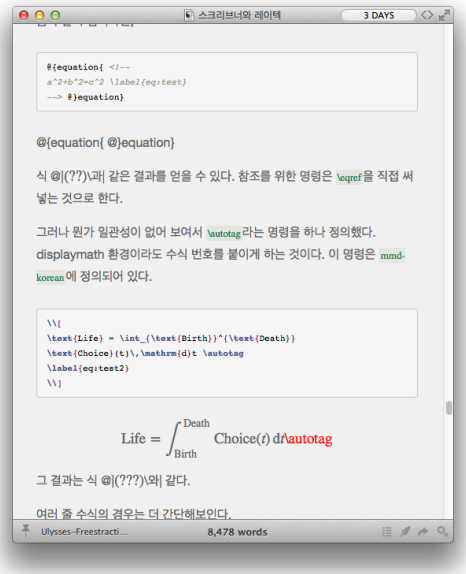

그림 6.1: marked2 미리보기

라텍 수식 작성에 익숙하지 못한 분을 위한 괜찮은 앱으로 Daum Equation Editor를 활용하는 것도 좋다. Mac App Store에서 다운로드하거나 Chrome Store에서 Chrome 플러그인을 설치하거나 하여 수식을 마치 MathType처럼 메뉴를 통하여 입력하고 라텍 코드를 얻을 수 있다.

6.2 번역하기

특히 "번역"에 있어서 스크리브너 작업 환경은 매우 유용하였다. 이에 관한 몇 가지 사항을 지적해두려 한다.

1. 번역할 대상인 원문을 Research 폴더에 가져다둔다.
2. QuickReference로 해당 원문을 열고 화면 한 쪽에 가져다 둔다.
3. 스크리브너 작업창에 번역문을 입력한다.
4. Mac OS X Mavericks의 사전 기능을 최대한 활용한다. 원문의 찾을 단어를 선택한 다음 right click → Writing Tools → Look up in dictionary를 누르면 맥 오에스의 사전이 열린다. 영한 사전과 위키백과가 있으므로 번역에 확실

히 참고가 된다. 메뉴에서는 Edit → Writing Tools → Look up in dictionary 로 접근 가능하다.

6.3 문서와 자료의 유지, 보존, 생성

이 절에서 설명하려는 것은 File 메뉴에서 찾을 수 있는 Sync with External Folder 기능에 대한 것이다.

특히 dropbox같은 클라우딩 툴을 이용하여 작업하는 경우라면 Sync할 외부 폴더를 dropbox 내의 특정 폴더로 한다. 이후로는 작업을 저장할 때마다 해당 폴더에 동기화가 이루어진다.

동기화할 파일 포맷을 결정할 수 있는데, 일반적으로 plain text면 충분하다. dropbox를 통하여 해당 파일을 여러 사람과 공유할 수 있다. 스크리브너 파일 자체를 공유하는 것은 바람직하지 못하다. 왜냐하면 해당 파일을 읽을 수 없을 사람도 있기 때문이다.

dropbox를 이용하여 공동 작업을 하는 경우라면, 스냅샷 기능을 적극적으로 활용해야 한다. 쌍방간에 수정이 이루어졌을 때, 이것을 추적하고 변경 사항을 악셉트하고 하는 것이 좀더 수월하게 이루어진다. svn같은 버전 관리 시스템에 비할 바는 아니지만 간이 버전관리라고 하기에 충분하다.

6.4 스냅샷과 변경 추적

한편, 스크리브너의 강력한 기능 중 하나가 스냅샷이다. 사실상 스크리브너를 주목한 이유 중의 하나가 스냅샷을 통한 (비교적 강력한) 버전 관리 때문이었다.

스냅샷은 도큐멘트에 대해서만 찍을 수 있다. 바인더에서 여러 문서를 선택하고 스냅샷을 찍는 것이 가능하다. 폴더는 스냅샷이라는 개념이 없다.

메뉴의 Documents → Snapshots → Take Snapshot을 찾거나 Cmd-5[3] 단축키로 바로 스냅샷을 찍을 수 있으며 스냅샷에 별도의 라벨을 붙이려면 Shift-Cmd-5 단축키로 라벨명을 적어주면 된다.

인스펙터 창에서 스냅샷을 볼 수 있다. compare한 후 화살표 버튼을 눌러서 변경 사항을 추적한다. roll back을 이용해서 스냅샷 시점으로 되돌릴 수 있다.

[3] 5자가 S자 비슷하게 생겼다.

스냅샷과 별도로 백업 기능이 있다. File → Back Up에서 찾을 수 있는데 이것은 말 그대로 문서의 현재 상태를 저장하는 것이다. 디폴트는 자동 백업이 이루어지는 것이지만 Back Up to라는 메뉴를 선택하여 현재 도큐멘트의 사본을 별도로 저장해두는 것도 가능하다. 그러나 자동 저장, 스냅샷이 있는 상태에서 문서의 백업 기능이 크게 필요하지 않다. 백업 사본이 많아지는 것은 파일 관리에 있어서 결코 권장할 일이 아니다. 관리해야 하는 파일은 적을수록 좋다. 백업을 강조하는 것은 정보의 유실을 염려한 것인데 스크리브너는 그에 대한 대비가 있기 때문이다.

6.5 통계

문서의 분량을 계산할 수 있다.

1. 도큐멘트의 하단에 단어수와 문자수가 표시된다. 편집하면 지속적으로 변한다.
2. 메뉴의 Project → Text Statistics를 선택하면 현재 작성 중인 도큐멘트에 대한 통계를 얻을 수 있다. 단어 수, 글자 수(스페이스를 계산에서 제외한 것), 문단 수, 단어의 사용 빈도 등을 볼 수 있다.
3. 메뉴의 Project → Project Statistics는 프로젝트 전체에 대한 통계를 보여준다.

한편, 메뉴의 Project → Project Target을 통해서 열 수 있는 "프로젝트 타겟"이라는 것은 도큐멘트와 프로젝트 전체에 대해 미리 분량을 정해놓고 진척과정을 체크할 수 있다. 정해진 분량의 글(기사, 논고, 일일드라마 등)을 써야 하는 경우에 요긴할 것이다.

6.6 키워드의 활용

도큐멘트마다 키워드를 붙일 수 있다. 키워드를 잘 활용하여 문서를 효율적으로 관리하자. Ctrl-Option-Cmd-J 단축키로 인스펙터의 키워드 창을 열어 추가할 수 있다. Project → Show Project Keywords (단축키 Shift-Cmd-K)를 열면 현재 프로젝트의 모든 키워드들이 보인다. Search하여 특정 키워드가 부여된 도큐멘트들을 모아서 관리할 수 있다. Show Project Keywords는 인스펙터 창에서도 열 수 있다.

6.7 아이패드와의 연동

아이패드용 스크리브너는 아직 나오지 않았다. 스크리브너 자체는 Simplenote[4]와 Index Card[5] 앱과의 Sync를 지원한다.

이 동기화 지원은 텍스트에 한정된다. 원래부터 simplenote 자체가 텍스트 위주의 앱이었고 index card 역시 마찬가지다. 그러나 아이패드를 이용한 문서 작성이라는 것이 부분적 수정, 아이디어의 메모와 같은 것이 위주가 될 것이기 때문에 원본을 손상하지 않는다면 이 정도로도 의미있는 기능이라고 생각한다.

6.8 한글 코드 문제

한글 문서 작성과 관련하여 한글 자체의 문제 몇 가지를 이해해두어야 한다.

1. 프로젝트 이름을 별도로 지정하지 않으면 파일 이름이 프로젝트 이름이 되는데 한글일 경우 첫가끝 코드이다. 이것은 다음과 같은 문제를 일으킨다.
   - `MultiMarkdown → PDF`에서 pdflatex이 실행되면 컴파일에 실패할 수 있다.[6]
   - xelatex의 경우 지정된 mainhangulfont가 첫가끝을 지원하지 않을 때 문서 타이틀이 네모박스로 나타날 수 있다. 라텍 스타일의 폰트를 첫가끝 지원하는 것으로 수정해도 되지만, Project Meta-data에서 Title을 적어주거나 Compile의 Meta-data에서 Title을 명시적으로 지정하는 것이 좋다.[7]
2. 라벨에 한글 문자가 쓰인다. 예를 들어 다음 경우를 생각하자.

   ```
   # 서론 #
   ```

   이 코드는 다음과 같이 변경된다. (base header level=2)

   ```
   \chapter{서론}
   \label{서론}
   ```

[4] http://simplenoteapp.com

[5] http://www.denvog.com/app/index-card/

[6] kotex-midkor라는 패키지를 설치하면 pdflatex에서도 첫가끝 코드를 에러없이 컴파일할 수 있으나 이 패키지는 별도로 설치하여야 한다.

[7] 한글 파일 이름을 쓸 수 없는 윈도우즈라면 이 부분에서 많은 어려움을 겪어야 한다. 맥 OS에서는 적어도 한글 파일 이름 때문에 컴파일이 되지 않는 일은 없다.

pdflatex은 이것을 적절하게 처리할 수 없다. `\label`의 인자로 한글이 올 수 없기 때문이다. pdflatex도 안전하게 컴파일하게 하려면 모든 장절 표제(즉 도큐멘트와 폴더의 타이틀)와 상호참조를 위한 그림, 표 등에 강제 라벨을 붙여야 한다. 라벨과 상호참조 (section 3.20) 절을 참고하라.

이 문제는 컴파일러로 xelatex을 사용하면 해결된다.

참고로, 스크리브너의 Compile for 옵션에서 `MultiMarkdown → PDF`를 실행하면 "use xelatex" 메타데이터가 있어도 pdflatex이 실행된다. 한편 Compile을 `MultiMarkdown`으로 하여 얻은 mmd 파일에 대하여 `mmd2pdf` 스크립트를 실행하면 "use xelatex"이 있을 경우 xelatex을 실행한다. 즉 "use xelatex" 메타데이터는 `mmd2pdf` 스크립트를 위한 것이다.

일반적으로 말해서 `MultiMarkdown → LaTeX`으로 얻은 `.tex`을 직접 (xelatex으로) 컴파일하는 것이 이 문제에 대한 해결책일 것이다.

CHAPTER 7

결론: 문서 작성의 생산성

지금까지 서술한 "스크리브너, 멀티마크다운을 이용한 라텍 문서 작성"을 위해서 라텍을 전혀 몰라도 된다고는 말하지 않았다. 오히려 라텍을 잘 알면 알수록 더 라텍에 신경쓰지 않는 작업이 가능하다. 그러나 적어도 잘 정비된 스타일 파일들과 스크리브너 템플릿이 있다면 라텍을 전혀 모르는 저자라도 라텍 문서를 생산할 수 있는 것임에도 틀림없다.

라텍의 강력한 조판 기능을 간단한 마크업을 통해서 이용한다는 이 발상이 더 나은 문서 생활과 연구 저작 활동에 도움이 되기를 바라며 글을 마친다.

그림 7.1: Do you want to build a snowman?

참고 문헌

[1] Allin Cottrell (2003), 김강수 역 (2013), "워드 프로세서, 멍청하고 비효율적인 도구," 온라인 문서. 번역문 url: http://doeun.blogspot.kr/2013/12/blog-post.html, 원문 url: http://ricardo.ecn.wfu.edu/∼cottrell/wp.html.

[2] Wikipedia, "Lightweight markup language," http://en.wikipedia.org/wiki/Lightweight_markup_language.

[3] Fletcher Penney, "멀티마크다운이란 무엇인가?," 웹문서. 원문 url: http://fletcherpenney.net/multimarkdown/. 번역문 url: http://doeun.blogspot.kr/2014/02/fletcher-penney-multimarkdown.html

[4] 김영욱, "TEX 사용에서 주의할 점," 온라인 문서. http://faq.ktug.org/faq/gromov?action=download&value=tex_u_ywk_xe.pdf.