예제

지난번에 작성한 `\DWalk` 명령이 있었다. 이것을 배포가능한 파일 묶음으로 만들어 보아라.

1 Packages

LaTeX을 처음 배울 때 class와 package를 알게 된다. class는 문서에서 단 하나만 지정할 수 있고 패키지는 여러 개를 use할 수 있다고 하였다.
자신이 작성한 expl3 명령을 다른 사람도 사용할 수 있게 하려면 이를 패키지로 배포하여야 한다. 굳이 배포하지 않더라도 반복 사용하는 코드들을 모아서 개인적 패키지로 만들어두는 사람들이 많다.
패키지라는 개념은 LaTeX 2_ε에서 처음 도입된 것이다. Leslie Lamport가 만든 LaTeX에는 클래스와 패키지의 구별이 없었다. 그래서

```
\documentstyle[a4,fancyheadings,psfig]{article}
```

이런 식으로 문서를 시작했는데 그러면 `article.sty`와 `a4.sty`, `fancyheadings.sty` 등을 차례로 로딩하는 방식이었다. 그러다보니 중복정의, 코드의 충돌 등 심각한 문제가 발생하기 시작했고 이를 해결하기 위해서 클래스와 패키지를 구분하게 한 것이다.
클래스에 대한 이야기는 다른 곳에서 하기로 하고 여기서는 패키지만 문제삼겠다.
Expl3 시대에 와서 패키지는 크게 두 종류로 구분된다. 하나는 전통적인 LaTeX 2_ε 패키지이고 다른 하나는 "Expl Package"라는 것이다. 본질만을 보자면 둘 사이에 큰 차이는 없지만 Expl Package는 LaTeX 2_ε 패키지가 가지지 못하는 몇 가지 속성을 더 가지고 있다. 우리는 Expl Package에 대해서만 다룰 것이고 LaTeX 2_ε 패키지 작성 방법에 대해서는 더 말하지 않을 생각이지만 아래 몇 가지를 지적하겠다.

1.1 LaTeX 2_ε 패키지에 대한 몇 가지

sty 파일 패키지 파일은 `.sty` 확장명을 갖는다. 이 파일을 문서에 불러올 때는 오직 preamble에만 올 수 있는

```
\usepackage[<options>]{<name>}
```

이라는 문장을 쓴다.
`\RequirePackage`라는 문장도 가끔 볼 수 있을텐데, 이것은 주로 패키지 제작자를 위한 명령으로서 `\usepackage`가 `\documentclass` 명령 이전에 올 수 없다는 제한을 없앤 것이다. 이 명령을 사용자가 사용하지 못하게 한 이유는 `\documentclass` 이전에 오는 코드들이 클래스 코드와 충돌할 때에 이를 해결할 능력을 문서작성자에게 요구할 수 없기 때문으로서, 일상적인 LaTeX 작업에서는 이 명령을 쓰지 않는다.
`\usepackage`는 해당 패키지를 불러들이면서 다음과 같은 일을 추가로 한다. (1) 패키지를 등록하여 이후 같은 패키지가 또 불릴 적에 이를 무시할 수 있게 한다. (2) 패키지를 불러오기 직전에 `\makeatletter`를 두고 패키지 불러오기가 끝난 시점에 `\makeatother`를 삽입한다.

패키지의 identity 예를 들어 `simpletest.sty`라는 파일이 있다고 하자. 이 파일의 제일 처음에 보통 다음과 같은 두 행이 있음을 볼 수 있다.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{simpletest}[2019/07/14 version 1]
```

첫 줄의 `\NeedsTeXFormat{LaTeX2e}`는 이것이 LaTeX 2_ε 패키지임을 의미한다. LaTeX 2_ε가 아닐 때 에러 메시지를 출력한다. 지금 LaTeX 버전이 몇십년째 LaTeX 2_ε라서 사실 이 선언이 별 필요없는 때도 많고 귀찮아서 생략하는 사람들도 생겨나고 있지만 웬만하면 적어두는 것이 좋다.
`\ProvidesPackage` 명령이 중요하다. 이것은 LaTeX 시스템에게 이 패키지의 아이덴티티를 전달하는 역할을 한다. 이 가운데 중요한 것은 "패키지의 이름"과 "날짜" 두 가지이다. 여기서 선언된 "패키지 이름"이 동일하면 그 패키지를 중복 로딩하지 않는다. 한편 "특정 날짜 이전에 작성된 패키지 파일은 로딩하지 말라"는 식으로 명령을 줄 수 있는데 아주 특별한 경우에 쓰인다.
가끔 파일 자체의 이름과 패키지가 선언한 이름이 다를 때가 있다. 이 때는 LaTeX이 이 패키지를 로드할 때 뭔가 이상하다는 메시지를 콘솔에 뿌려준다. 동작에 큰 이상이 있는 것은 아니지만 에러메시지가 불쾌하고 중복 로딩에 착오가 생길 가능성도 있으므로 이 둘은 일치시켜 두어야 할 것이다.

패키지의 옵션 패키지의 옵션은 브래킷 안에 쉼표로 분리된 리스트로 부여한다. 어떤 옵션이 패키지에게 주어졌는지를 검사하는 패키지 내부 명령으로 `\ProcessOptions` 류의 명령이 사용된다. 옵션이 전달되었을 때 그 옵션에 따라 어떤 동작을 취할 것인가는 전적으로 패키지 내부의 문제이다.
`\PassOptionsToPackage`라는 명령은 이후에 `\usepackage`할 같은 이름의 패키지에게 미리 정해진 옵션을 넘기라는 명령이다.

dtx라는 파일 옛날에는 LaTeX 패키지가 전부 `.dtx`와 `.ins` 파일로 배포되었다. 이 방법은 아직도 쓰이지만 굳이 알아둘 필요가 크지는 않아서 이게 뭔가만 간단히 설명하고 지나간다. `.dtx` (Documented LaTeX) 포맷은 *code*와 *document*를 하나로 결합한 파일이다. `cweb`이 일반 프로그래밍 언어에 대해서 동작하는 "문학적 프로그래밍" 툴이라면 `.dtx`는 LaTeX의 문학적 프로그래밍이라고 할 수 있다. `.dtx` 문서로부터 *code* 부분만 추려내기 위해서 `.ins`라는 docstrip 파일을 제공한다. `latex`을 `.ins`에 적용하여 컴파일하면 *code*에 해당하는 부분이 파일(주로 `.sty`)로 풀려나오고 `.dtx` 자체에 대해서 `latex`을 적용하면 "문서"가 컴파일되어 나온다.
이 포맷에 관심이 있다면 Scott Pakin의 "An Introduction to writing `.ins` and `.dtx` files"라는 글(*TUGboat*19:2 (2008))과 docstrip 패키지 문서를 참고하라. 우리는 `.sty`를 `.dtx`로 묶는 것까지 다루지는 않겠다.

1.2 패키지를 만들기 전에

패키지를 꾸릴려면 적어도 다음과 같은 것이 준비되어 있어야 한다. 우리가 dummy라는 이름의 패키지를 만들기로 하고

(1) 패키지로 조성된 파일 자체. `dummy.sty`.

(2) 패키지 사용안내 문서. 적어도 문서의 소스와 pdf가 함께 제공되어야 한다. `dummy-doc.tex`, `dummy-doc.pdf`. `texdoc`이 이 문서를 쉽게 찾아야 하므로 패키지 자체의 이름과 일치시켜 두는 것이 좋고 대부분 `-doc`이 붙는 정도가 허용된다. 패키지 제작자의 입장에서 세상에서 가장 귀찮은 일이 문서를 만드는 것이다. 그러나 사용법 문서가 없는 패키지는 하등 아무짝에도 쓸모가 없다. 그건 그냥 쓰레기. 만약 문서가 특별한 그림 파일 따위를 이용하고 있다면 그런 것도 함께 묶어서 소스를 컴파일하여 동일한 문서를 얻을 수 있게 해야 한다. 예전에는 패키지 문서에 "구현"을 상세히 서술하였는데(dtx 문서의 성격상 코드 해설이 위주가 되는 경향을 띨 수밖에 없었다) 이것은 개발자에게 매우 중요한

정보이다. 그러나 당장 필요한 것은 사용자에게 필요한 사용법 정보일 것이다. 이것만이라도 문서로 만들어두어야 한다.

(3) 저작권. LaTeX 패키지는 보통 LPPL이라는 copyright로 배포한다. 저작권 표시는 반드시 남기는 것이 좋다.

(4) 상세한 주석. 만약 패키지를 유지-보수할 의사가 있다면 자신이 작성한 코드에 일일이 세밀한 주석을 달아두어라. 이 코드를 왜 이 모양으로 짰는지 적어두지 않으면 수습불가능한 상황이 되고 결국 업데이트마저 불가능하게 될 것이다. 이 주석은 패키지 파일 안에 주석문으로 써넣어둔다.

1.3 Expl Package

Expl Package의 Identity Expl 패키지는 먼저 다음과 같이 시작한다. `expl3` 패키지를 먼저 로드하기 때문에 `\NeedsTeXFormat{LaTeX2e}` 같은 라인은 불필요하다.

```
\RequirePackage{expl3}
\ProvidesExplPackage
    {dummy}
    {2017/02/13}
    {0.001}
    {a dummy package for learning expl3 programming}
```

`\ProvidesExplPackage` 명령은 네 개의 인자를 갖는다. ① 패키지의 이름, ② 날짜, ③ 버전, ④ 패키지 설명.
이 선언이 불리게 되면 그 이후부터 해당 파일의 끝까지 expl syntax 상태가 된다. 즉 `\ExplSyntaxOn`과 `\ExplSyntaxOff`를 별도로 지정하지 않아도 된다.
(Expl Package가 아닌 LaTeX 2_ε 패키지에서는 `\RequirePackage{expl3,xparse}`를 선언하고 나서 expl3 코드를 `\ExplSyntaxOn`과 `\ExplSyntaxOff` 사이에 넣어야 한다.)

Expl Package의 옵션 실제로 LaTeX 2_ε 패키지 제작의 가장 어려운 점은 옵션을 처리하는 것이었다. 보통 간단한 단어로 주어지는 옵션이야 어떻게든 할 수 있지만 LaTeX 2_ε가 발달하면서 점점 수요가 많아진 <*key*> = <*value*> 옵션을 처리하는 것이 제법 난관이었는데 그러다보니 keyval이니 xkeyval이니 하는 복잡하기만한 추가 패키지가 요구되는 그런 상황이었던 것이다.
expl3를 써야 하는 이유 중의 하나가 이런 <*key*> = <*value*> 옵션을 너무나 간단하게 처리할 수 있다는 것이다. l3keys2e라는 패키지는 expl3의 keys 데이터를 패키지의 옵션으로 사용할 수 있게 만들어준다. 다음 보기를 보자.

```
\RequirePackage{l3keys2e}

\keys_define:nn { dummymain }
{
    clear  .bool_set:N = \g_clear_bool,
    print  .tl_set:N = \g_print_tl
}

\ProcessKeysOptions { dummymain }
```

이것이 전부다. 이제

```
\usepackage[print={Please Help Me}]{dummy}
```

라고 하면 이 패키지 내부에서 `\g_print_tl`의 값이 주어진 문자열이 될 것이다.

1.4 패키지 작성 실전: esgdwalk 패키지

이름 짓기 패키지를 만들기로 하였다면 그 이름을 지어주어야 한다. 제작자의 입장에서야 짧고 간단하고 기억하기 쉬운 패키지 이름과 명령 이름을 짓고 싶겠지만 웬만한 단어들은 이미 다들 써먹었을 것이 틀림없어서 그게 그렇게 간단하지 않다. 특히 사용자 인터페이스 함수는 물론이고 내부 함수의 이름도 어디의 누군가가 이미 써버린 것이 확실할 정도로 간단한 이름으로 해두면 곤란하다. 내부 함수 이름은 특히 어디에서도 중복되지 않도록 작성해두는 것이 좋다. 우리가 인터페이스 함수 이름을 간단히 `\walk`이나 `\dwalk`이라고 이름짓기 꺼리는 이유는 거기에 있다. 최대한 양보한 것이 대문자를 섞어쓰는 `\DWalk`이었던 것이다. 예컨대 `\randomwalk` 이런 좋은 이름을 누군가 선점했을까 그렇지 않았을까? 혼자서 쓰는 패키지는 아무래도 문제 없겠으나 일단 "배포"하기로 했다면 이런 점을 잘 고려하여야 한다.
조금 다른 이야기지만 얼마 전까지 (또는 지금도) TEX으로 문서를 만드시는 분들은 자기 편하자고 예를 들면 `\def\v#1{\vec{#1}}` 이런 식으로 해놓고 쓰는 경우가 정말 정말 많았다. 그런 원고를 받아서 편집하는 사람의 입장에서 이것은 말할 수 없이 짜증나는 일인데, `\v`는 ě할 때 쓰이는 표준 LATEX 명령인 것이다. 이걸 `\def`해버리면 어쩌라는 말인가? 게다가 두 분 이상의 원고를 받아서 한 권의 책으로 묶는 상황에서 이것은 수많은 재정의 충돌을 일으킨다. LATEX 2_ε 이후로 사용자는 preamble에서 `\def` 대신 `\newcommand`를 쓰라고 그렇게 강조를 해도 잘 말을 듣지 않는 사람들이 많다. 핵심은 "남들이 이미 썼을 만큼 짧고 쉬운 명령은 사제로 만들어 쓰지 말라"는 것이다.
패키지의 이름은 더 까다로운데 이쪽은 너무 길어져도 곤란하기 때문이다. 물론 옛날같은 `8.3` 제한이 있는 것은 아니지만.
한때 제작자 이름을 앞에 붙이는 방식의 명명법이 유행한 적이 있는데 CTAN에서 한참 전부터 패키지 이름에 "자기 이름 첫글자를 붙이지 말라"고 하고 있다. 그것만 보아서는 도무지 어디에 쓰는 물건인지 알기 어렵게 하는 원인이었기 때문인 듯하다.
지금 제작해보려 하는 패키지의 이름은 `esgdwalk`으로 하려고 하는데 위의 기준에 맞추어보면 앞에 붙은 `esg`라는 표현은 범용 패키지로 배포하려 할 때는 좋은 명명법이 아니라고 하겠다.

시작 부분의 주석 보통 이 위치에는 이 파일의 이름과 저작권자, 저작권 및 기타 정보 (간략한 사용법 등)를 주석문으로 기록해둔다. 다음과 같이 시작한다.

```
%
% file: `esgdwalk.sty`
%
% (c) 2019 Nova de Hi.
%
% A short sample package for Expl3 Study Group of KTUG
%
% ======= LPPL =======
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3 or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Nova de Hi.
%
```

여기서는 라이센스를 LPPL로 했는데, 원한다면 GPL이나 BSD Lisence나 Creative Commons License 등을

채택할 수 있다. 아니면 전적으로 사유(proprietary)로 선언하고 필요하면 돈을 내고 사라고 써놓을 수도 있겠지.

Expl Package 선언 다음과 같이 선언한다. expl3를 require하는 선언은 위의 주석문 바로 다음 행에 행을 띄지 말고 적어야 하는데 `\ProvidesExplPackage` 선언이 오기 전까지는 expl syntax가 아직 아니므로 여기를 띄어두면 "의도하지 않은 스페이스"가 들어갈 수 있기 때문이다. 대부분의 경우 별 문제 없으나 습관을 이렇게 들여두는 쪽을 추천한다.

```
\RequirePackage{expl3,xparse,l3keys2e}
\ProvidesExplPackage
    { esgdwalk }
    { 2019/08/03 }
    { 1.0 }
    { a sample expl3 package }
```

이 선언이 있은 이후부터 바로 `\ExplSyntaxOn` 되어 있다.

필수 패키지 이 패키지를 위하여 반드시 필요한 패키지를 로드한다.

```
\RequirePackage{tikz}
\RequirePackage{esgutil}
```

이 경우에 esgutil이라는 패키지가 배포판의 범용 패키지가 아니므로 이것을 어디에서 받을 수 있는지 설명하거나 아니면 (저작권의 문제가 없는 경우에) 함께 포함하여 배포하여야 한다.

옵션의 처리 옵션을 딱 하나만 주어보자. `[color=blue]`라는 식의 옵션을 받아들여서 선의 색상을 주도록 하겠다.

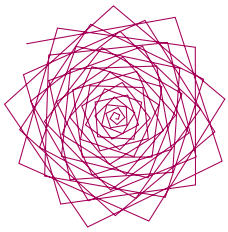
```
%%% package options
\keys_define:nn { dwalkpackage }
{
    color    .tl_set:N = \g_coloropt_tl
}

\ProcessKeysOptions { dwalkpackage }
```

이제 패키지 옵션은 `\g_coloropt_tl`로 넘어온다. 디폴트 색상을 주고 싶으면 `\ProcessKeysOptions` 명령 이전에 `\keys_set:nn` 해두면 될 것이다.

메인 코드 esg006에서 이미 만들어 본 `\DWalk` 코드를 그대로 가져오면 된다. 거기에서 또 keys 정의가 있는데 명령의 인자를 찾아내는 key를 module명을 dwalk으로 하였다. 이것이 패키지의 옵션을 받아오기 위한 keys의 module명과 달라야 할 것이다.
메인 코드는 첨부한 esgdwalk.sty의 소스 코드를 참고하여라. 이전에 했던 그대로이고 단지 색상 관련 코드만 추가하였다.
파일의 마지막에 `\endinput`을 두어서 파일의 끝임을 알려도 좋다.

문서 작성 이 패키지가 사용자에게 전달되었을 때 사용자는 패키지의 코드를 보는 것이 아니라 설명 문서를 원한다. 문서를 간단히라도 만들어보자. `esgdwalk-doc.tex`과 `esgdwalk-doc.pdf`가 문서에 해당한다.
여기까지 되면 다 된 것이다. 이 세 개의 파일 `esgdwalk.sty`, `esgdwalk-doc.{tex,pdf}`를 하나의 압축파일로 묶어서 배포한다.
`\usepackage[color=red!30!violet]{esgdwalk}`을 preamble에 두고 `\DWalk` 명령을 사용해보았다.

```
\DWalk{angle=82,forward=0.02+0.02,walk=100}
```



패키지의 배포 LaTeX 패키지는 CTAN을 통하여 배포한다. 자신이 작성한 패키지가 여러 사람에게 쓸모가 있고 버그가 없으며 사후 대응 (버그 수정, 업데이트) 에 자신이 있다면 CTAN에 업로드하여도 좋다.
한국적 상황에 맞는 패키지들은 KTUG Private Repository를 통하여 배포할 수도 있다.

연습문제

지금까지 자신이 작성한 함수와 명령 중에서 마음에 드는 것을 모아 자신의 이름을 붙여서 패키징하고 이를 KTUG Wiki의 “Expl3 Study Group” 페이지에 업로드하여라.

Nova de Hi.