# Yet Another Guide to How to Define Macros

Yi, Hoze and Nova de Hi

2012, 2015

**요 약**

이 짧은 글은 이호재(Yi, Hoze)의 "A Guide to How to Define Macros," 2012에 소개된 라텍 매크로 작성 방법을 소개하고 거기에 해당하는 LaTeX3 (expl3)의 매크로 정의 방법을 주석으로 붙여서 만든 것이다. 굳이 expl3 코딩 방법에 대하여 관심을 갖지 않더라도 원본의 내용만으로도 라텍의 매크로 작성 기법에 대하여 이해를 도울 수 있을 것으로 기대한다. expl3 부분은 Nova De Hi가 작성하였다. expl3의 기능을 소개하는 것이 아니라 원래의 코드를 expl3로 충실히 번역(재현)하는 것을 목표로 한 것임을 감안하여 주시라.

## 1 `\makeatletter`, `\makeatother`

```
\makeatletter
....
\makeatother
```

사용자가 문서를 작성하는 document 환경 내에서 at-문자(@)는 매크로 이름에 쓰일 수 없다. 그런데 LaTeX에서 "내부 명령" 즉 사용자가 직접 접근할 수 없지만 매크로 정의를 위해 필요한 매크로들을 만들기 위해 @을 사용하는 것이 관행이다.
이 문자가 매크로 명령 이름에 사용될 수 있도록 하라는 것이 `\makeatletter`이고 원래대로 되돌리라는 것이 `\makeatother`이다. 모든 스타일 파일은 이 명령을 지정하지 않아도 `\usepackage`할 때 @을 매크로에서 사용할 수 있도록 되어 있다. 그러므로 파일의 확장자가 `.sty`라면 그 파일 내의 `\makeatletter`, `\makeatother`를 모두 제거하는 것이 좋다.

## 1 `\ExplSyntaxOn`, `\ExplSyntaxOff`

```
\ExplSyntaxOn
....
\ExplSyntaxOff
```

Expl3 문법으로 매크로를 작성할 때 `\ExplSyntaxOn` 상태이어야 한다. 이 상태에서는 (1) 모든 스페이스가 무시되고, (2) 언더스코어 문자(_)와 콜론 문자(:)가 매크로 정의에 사용된다. 이 환경의 종료는 `\ExplSyntaxOff`이다.
아래 모든 Expl3 예제는 모두 `\ExplSyntaxOn`과 `\ExplSyntaxOff` 사이에서 정의되는 것이다. 별도로 이를 표시하지 않는다.
expl3 패키지로 작성된 경우에는 이 구문법 On/Off를 별도로 지정하지 않아도 된다. 그러나 일반적인 LaTeX $2_\varepsilon$ 패키지에서는 위의 명령을 지정해야 expl3 구문을 쓸 수 있다. 스페이스(공백)는 명시적으로 지정해야 한다. 틸데 문자(~)나 `\space` 매크로를 사용한다.

아래 예제에서 @문자가 매크로에 사용된 경우, 이것을 preamble에서 적용하려면 \makeatletter와 \makeatother가 필요하다. 이 예제 문서에서는 별도로 이를 표시하지 않는다.

## 2 \def, \newcommand

```
\def\foo#1#2{... #1 ... #2}
```

plain TEX의 매크로 정의(*def*ine). 이밖에도 \xdef, \edef, \long\def 등이 있지만 여기서는 주로 LATEX의 방식을 문제삼을 것이라서 더 언급하지 않았다.

```
\newcommand\foo[2]{... #1 ... #2}
\renewcommand\foo[2]{... #1 ... #2}
\newcommand*\foo[2]{... #1 ... #2}
\renewcommand*\foo[2]{... #1 ... #2}
```

\newcommand와 \def의 차이는 대략 다음과 같다.

- 기본적으로 \long으로 정의된다. 즉 \par를 인자로 받을 수 있다. \par가 필요없는 (짧은) 명령을 정의할 때는 별표(*)를 붙인다.
- 같은 이름의 control sequence가 이미 정의되어 있는지를 체크한다. 만약 같은 이름이 이미 있고 그것을 수정하려는 것이라면 \renewcommand를 사용해야 한다. 이를 통하여 \def을 쓸 때 일어날 수 있는 "덮어쓰기 정의"의 위험을 피할 수 있다.
- 인자의 개수를 미리 지정한다. \def\foo#1#2는 \newcommand[2]에 해당한다. 명령의 정의부에서 인자를 #1, #2로 쓰는 것은 동일하다.

```
\newcommand*\mysymfont{%
   \fontspec{HCR Batang LVT}%
   \hangulfontspec{HCR Batang LVT}%
```

expl3 문법을 사용하기 위해서 preamble에 \usepackage{expl3,xparse} 선언을 두는 것이 좋다. oblivoir의 경우에는 이 문장이 없어도 expl3 문법을 쓸 수 있다.

## 2 \cs_new, \NewDocumentCommand

```
\cs_new:Npn \foo:n #1 { ... #1 ... }
\cs_new:Npn \foo:nn #1 #2 { ... #1 ... #2 ... }
```

expl3은 "함수"와 "변수"를 구분한다. 위의 예제는 "함수"를 정의하는 것인데, 함수 이름에는 반드시 "인자 지정자"가 붙는다. \cs_new:Npn의 :Npn이나 여기서 정의한 \foo의 :nn 부분이 인자 지정자이다. 인자 지정자의 종류에 대해서는 ??페미지 제?? 절을 참고. expl3에서는 함수 이름 자체에 어떤 종류의 인자 몇 개를 취하는지에 대한 정보가 포함되어 있다.
expl3의 함수를 사용자가 직접 문서에서 호출할 수 없다. 사용자 인터페이스 명령을 xparse 패키지의 \NewDocumentCommand 명령으로 제공해야 한다. 사용자 인터페이스 명령의 이름을 지을 때 언더스코어나 @문자 등을 사용하지 않도록 유의한다.

```
\usepackage{xparse}
\NewDocumentCommand \foo { m } { ... #1 ... }
```

{ m } 부분이 이 명령의 인자를 지정하는 부분이다. m은 "일반 유형 인자"를 의미하고 들어오는 그대로의 토큰열을 명령 \foo에 넘겨준다. 여기서는 일반 유형 인자 한 개를 받는다는 것이다. 이것은 명령 정의부에서 #1에 대응한다. 인자 지정자의 종류는 제?? 절을 볼 것. \par를 포함하는 긴(long) 인자라면 { +m }로 한다.
ExplSyntax 영역 내에서 빈 칸은 전부 무시되므로 소스 코드를 읽기 좋게 적당히 띄어쓰는 것이 좋다. 행말의 EOL 문자를 없애기 위해서 % 표지를 붙이지 않아도 된다.

```
\cs_new:Nn \my_sym_font:
{
```

```
   \hanjafontspec{HCR Batang LVT}}
\newcommand*\mysym[1]{{\mysymfont\char"#1}}
\newcommand*\mysmiley{\mysym{263A}}
```

\LARGE \mysym{2639} \mysliley ☹ ☺

### 3 \providecommand

```
\providecommand\foo[2]{... #1 ... #2}
\providecommand*\foo[2]{... #1 ... #2}
```

```
\DeclareRobustCommand\foo[2]{... #1 ... #2}
```

풀리는 명령을 풀리지 않게 만들려면 etoolbox의 \robustify를 쓴다.

```
\usepackage{etoolbox}
\robustify{\TeX}
```

```
    \fontspec { HCR Batang LVT }
    \hangulfontspec { HCR Batang LVT }
    \hanjafontspec { HCR Batang LVT }
}

\NewDocumentCommand \MySym { m }
{
    \group_begin:
      \my_sym_font:
      \char"#1
    \group_end:
}

\NewDocumentCommand \MySmilie { }
{
    \MySym { 263A }
}
```

\LARGE \MySym{2639} \MySmilie ☹ ☺

### 3 \ProvideDocumentCommand

```
\ProvideDocumentCommand \foo { m +m }
{
   ... #1 ... #2
}
```

이밖에, xparse는 다음 명령을 제공한다.

(1) \DeclareDocumentCommand

(2) \RenewDocumentCommand

(3) \DeclareExpandableDocumentCommand

이렇게 하면 \section이나 \caption 명령 안에서 \protect해주지 않아도 이 명령이 풀리지 않는다.

## 4 Starred Commands

```
\newcommand\myemph{\@ifstar{\myemph@xii}{\myemph@xi}}
\newcommand*\myemph@xi[1]{\textcolor{blue}{\textit{#1}}}
\newcommand*\myemph@xii[1]{\textcolor{red}{\textit{#1}}}
```

\@ifstar는 \@ifnextchar*에 해당하는데 이 구문을 쓸 때 발생할 수 있는 코딩 오류를 피하기 위해 마련된 LaTeX 명령이다. 한편 expl3은 s형 인자에 대하여 \IfBooleanTF 검사를 함으로써 훨씬 쉽게 별표붙은 명령을 정의할 수 있다.

\myemph{Darth Vader}, \myemph*{Darth Vader}

*Darth Vader*, *Darth Vader*

## 5 Optional Arguments

## 6 One Optional Arguments

```
\newlength\myvert{}
\newcommand\myraise[2][\@empty]{%
  \ifx\@empty#1%
    \setlength\myvert{0ex}%
  \else
    \setlength\myvert{#1}%
  \fi
  \raisebox{\myvert}{#2}}
```

\large Darth Vader \myraise{\mysmiley} is a central character

xparse 방법으로는 기본적으로 풀리지 않는 명령으로 정의된다. 이 명령을 풀리는 명령으로 만들려면 \DeclareExpandableDocumentCommand로 정의한다.

## 4 \IfBooleanTF

```
\NewDocumentCommand \MyEmph { s m }
{
    \IfBooleanTF { #1 }
    {
        \textcolor{red}
    }
    {
        \textcolor{blue}
    }
    { \textit { #2 } }
}
```

\MyEmph{Darth Vader}, \MyEmph*{Darth Vader}

*Darth Vader*, *Darth Vader*

## 5 \IfNoValueTF

```
\NewDocumentCommand \MyRaise { o m }
{
   \IfNoValueTF { #1 }
   {
      \dim_zero:N \l_tmpa_dim
   }
   {
      \dim_set:Nn \l_tmpa_dim { #1 }
   }
   \raisebox { \l_tmpa_dim } { #2 }
}
```

```
\myraise[.25ex]{\mysmiley} in the Star Wars saga.
```

Darth Vader ☺ is a central character ☺ in the Star Wars saga.

## 7 Two Optional Arguments

두 개의 옵션 인자를 가진 명령을 정의하려면 twoopt 패키지의 `\newcommandtwoopt`를 이용할 수 있다. 세 개 이상의 옵션 인자에 대해서는 복잡한 정의를 피할 수 없다. 반면 expl3는 o형 인자에 대하여 `\IfValueTF` 또는 `\IfNoValueTF` 검사를 하는 것으로 쉽게 정의할 수 있다.

```
\usepackage{twoopt}

\newcommand*\ui[1]{\textsf{#1}\index{#1}}
\newcommandtwoopt{\menu}[4][\@empty][\@empty]{%
   \ifx#1\@empty
     \ui{#3} > \ui{#4}%
   \else
     \ifx#2\@empty
        \ui{#1} > \ui{#3} > \ui{#4}%
     \else
        \ui{#1} > \ui{#2} > \ui{#3} > \ui{#4}%
     \fi
   \fi
}
```

```
\menu{File}{New} \par
\menu[Format]{Systax Coloring}{LaTeX} \par
\menu[Search][Replace]{Files}{All}
```

File > New

Format > Systax Coloring > LaTeX

Search > Replace > Files > All

```
\large Darth Vader \MyRaise{\MySmilie} is a central character
\MyRaise[.25ex]{\MySmilie} in the Star Wars saga.
```

Darth Vader ☺ is a central character ☺ in the Star Wars saga.

## 6 `\IfNoValueTF`, `\IfValueTF`

여러 개의 옵션 인자를 잇대어 정의하는 경우, 각 옵션 인자를 나타내는 delimiter 문자 (디폴트는 `[ ]`) 사이에 공백이 있으면 파싱에 실패할 수 있으므로 주의를 요한다. 즉, document에서 아래 `\Menu` 명령을 쓸 때 `\Menu[abc]␣[def]␣{arg1}{arg2}`와 같이 옵션 인자 사이에 공백이 들어가면 에러가 발생할 수 있다.

```
\cs_new_nopar:Npn \my_ui:n #1
{
   \textsf { #1 } \index { #1 }
}

\NewDocumentCommand \Menu { o o m m }
{
    \IfValueT { #1 }
    {
       \my_ui:n { #1 } ~>~
    }
    \IfValueT { #2 }
    {
       \my_ui:n { #2 } ~>~
    }
    \my_ui:n { #3 } ~>~ \my_ui:n { #4 }
}
```

```
\Menu{File}{New} \par
\Menu[Format]{Syntax Coloring}{LaTeX} \par
\Menu[Search][Replace]{Files}{All}
```

File > New

Format > Syntax Coloring > LaTeX

Search > Replace > Files > All

## 7 *keys* datatype

```
\keys_define:nn { mytest }
{
   scale    .tl_set:N  = \my_scale_tl,
   raise    .dim_set:N = \my_raise_dim,
   showfilename .bool_set:N = \my_show_bool,
}

\cs_new:Nn \reset_mytest_keys:
{
  \keys_set:nn { mytest }
  {
     scale = 0.1,
     raise = -0.5ex,
     showfilename = false
  }
}

\NewDocumentCommand \lineFIG { o m }
{
   \IfNoValueTF { #1 }
   {
      \reset_mytest_keys:
   }
   {
      \reset_mytest_keys:
      \keys_set:nn { mytest } { #1 }
   }
   \raisebox { \my_raise_dim }
      { \includegraphics [ scale = \my_scale_tl ] { #2 } }
   \bool_if:NT \my_show_bool
```

## 8 Optional Keys

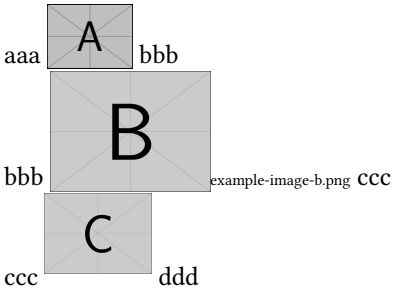
### 8.1 keycommand package

<key>=<value> 형식으로 =를 좌우에 두고 각각 *<key>*에 *<value>*를 할당하는 자료형을 명령의 인자로 구현하려면 xkeyval이나 keycommand 패키지를 이용한다. 반면, expl3에는 *keys*라는 데이터타입이 마련되어 있으므로 외부 패키지의 도움을 받지 않고 이런 형식의 자료 구조를 손쉽게 처리한다. expl 패키지에서 *keys* 자료 구조를 옵션으로 쓸 수 있도록 하는 l3keys2e 패키지도 준비되어 있다.

```
\usepackage{keycommand}

\newkeycommand\LineFig%
  [scale=0.1, raise=-0.5ex, bool showfilename=false][1]%
  {\raisebox{\commandkey{raise}}%
     {\includegraphics[scale=\commandkey{scale}]{#1}}%
   \ifthenelse{\equal{\commandkey{showfilename}}{1}}{{\tiny#1}}{}%
  }
```

```
aaa \LineFig{example-image-a} bbb \par
bbb \LineFig[scale=0.15, showfilename=true]{example-image-b.png}
    ccc \par
ccc \LineFig[raise=1ex]{example-image-c.png} ddd
```



### 8.2 xkeyval package

```
\usepackage{xkeyval}

\define@boolkey{foo}{bkey}[true]{\ifKV@foo@bkey ... \else ... \fi}
\define@key{foo}{akey}{ ... #1 ...}
\presetkeys{foo}{bkey=, akey=, ...}{}
\newcommand\foo[2][]{
\setkeys{foo}{#1}
... #2 ...
}

\foo[akey=, ...]{...}
```

## 9 Conditionals: xifthen package

### 9.1 Booleans

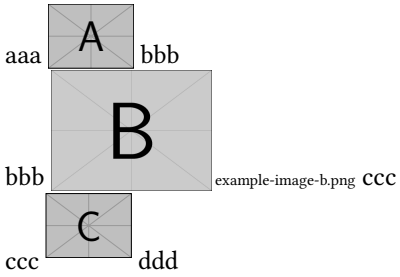
```
\ifx{\@empty}{#1} ... \else ... \fi
\newif\ifmyc@nd \myc@ndtrue \myc@ndfalse
\ifmyc@nd ... \else ... \fi
```

```
\newboolean{myc@nd}
\setboolean{myc@nd}{true/false}
\ifthenelse{\boolean{myc@nd}}{true}{false}
```

```
\newboolean{mycond}
\setboolean{mycond}{true}
\ifthenelse{\boolean{mycond}}{TRUE}{FALSE}
\setboolean{mycond}{false}
```

```
    {
       \begin{tiny} ~#2 \end{tiny}
    }
}
```

```
aaa \lineFIG{example-image-a} bbb \par
bbb \lineFIG[scale=0.15,showfilename=true]{example-image-b.png} ccc \par
ccc \lineFIG[raise=1ex]{example-image-c} ddd
```




## 8 Conditionals

### 8.1 *boolean* datatype

```
\bool_new:N \my_cond_bool

\NewDocumentCommand \setbooltest { m }
{
    \str_case:nnF { #1 }
    {
        { true }  { \bool_set_true:N \my_cond_bool }
        { false } { \bool_set_false:N \my_cond_bool }
    }
    {
        \bool_set_false:N \my_cond_bool
    }
```

```
\ifthenelse{\boolean{mycond}}{TRUE}{FALSE}
```

TRUE FALSE

### 9.2 strings

```
\ifthenelse{\equal{#1}{...}}{true}{false}
\ifthenelse{\equal{#1}{\str}}{true}{false}
```

```
\def\mystr{My String}
\ifthenelse{\equal{My String}{\mystr}}{Onaji}{Chigau}
```

Onaji

```
}

\NewDocumentCommand \booltest { }
{
    \bool_if:NTF \my_cond_bool
    {
        TRUE
    }
    {
        FALSE
    }
}
```

```
\setbooltest{true}
\booltest
\setbooltest{false}
\booltest
```

TRUE FALSE

### 8.2 *str* datatype

```
\NewDocumentCommand \StrComp { m }
{
    \str_if_eq:nnTF { #1 } { My~String }
    {
        Onaji
    }
    {
        Chigau
    }
}
```

```
\StrComp{My String}, \StrComp{MyString}
```

### 9.3 counters

```
\newcounter{mycnt}
\setcounter{mycnt}{3}
\renewcommand\themycnt{\Alph{mycnt}}
\themycnt
\addtocounter{mycnt}{2}
\themycnt
```

C E

```
\ifthenelse{\value{page} = \value{mycnt}}{true}{false}
\ifthenelse{\value{page} > 10}{true}{false}
\ifthenelse{\value{mycnt} < 10}{true}{false}
```

false false true

Onaji, Chigau

### 8.3 *int* datatype

```
\int_new:N \my_int
\int_set:Nn \my_int { 3 }
\int_to_Alph:n { \int_use:N \my_int }
\int_add:Nn \my_int { 2 }
\int_to_Alph:n { \int_use:N \my_int }
```

CE

```
\makeatletter
\NewDocumentCommand \checkpageoddeven { }
{
    \int_if_odd:nTF { \int_use:N \c@page }
    {
       ODD
    }
    {
       EVEN
    }
}
\makeatother
```

ODD

```
\int_compare:nTF { \my_int >= 10 }
{
    TRUE
}
{
    FALSE
}
```

FALSE

### 9.4 dimensions

```
\newlength\mylength
\setlength\mylength{10cm}
\ifthenelse{\lengthtest{\mylength < \textwidth}}{true}{false}
\ifdim \mylength > \linewidth true \else false \fi
```

true false

### 9.5 dimension of boxes

```
\newsavebox\mybox
\sbox\mybox{\hbox{Darth Vader}}

\settowidth\mylengthwd{\mybox}
\settoheight\mylengthth{\mybox}
\addtolength\mylengthth{\dp\mybox}

\raisebox{-\dp\mybox}{\rule{\mylengthwd}{\mylengthth}}%
\usebox\mybox
```

Darth Vader

### 8.4 *dim* datatype

```
\dim_new:N \my_dim
\dim_set:Nn \my_dim { 10cm }
\dim_compare:nTF { \my_dim < \textwidth }
{  TRUE  }
{  FALSE }
```

TRUE

### 8.5 *box* datatype

```
\box_new:N \my_box
\hbox_set:Nn \my_box { Darth~Vader }
\dim_set:Nn \l_tmpa_dim { \box_wd:N \my_box }
\dim_set:Nn \l_tmpb_dim { \box_ht:N \my_box }
\dim_add:Nn \l_tmpb_dim { \dim_eval:n { \box_dp:N \my_box } }
\raisebox { -\box_dp:N \my_box }
          { \rule { \l_tmpa_dim } { \l_tmpb_dim } }
\box_use:N \my_box
```

Darth Vader

### 8.6 *fp* datatype

floating point expression.

```
\fp_new:N \my_fp
\fp_set:Nn \my_fp { 3.14159265 }
\fp_add:Nn \my_fp { 1.4142135 }
\fp_use:N \my_fp,~
\dim_set:Nn \my_dim { \fp_to_dim:N \my_fp }
\rule{\my_dim}{5pt}
```

4.55580615, ■

### 8.7 *seq* datatype

```
\seq_new:N \my_seq
\seq_set_split:Nnn \my_seq {;} {a;b;c;d;e}
\seq_pop_left:NN \my_seq \l_tmpa_tl
\l_tmpa_tl \par

\seq_use:Nn \my_seq {|}
```

a

b|c|d|e

### 8.8 *clist* datatype

```
\clist_new:N \my_clist
\clist_set:Nn \my_clist { a, b, c, d, e }
\clist_use:Nn \my_clist { ;~ }
```

a; b; c; d; e

### 8.9 *property* datatype

```
\prop_new:N \my_prop
\prop_put:Nnn \my_prop { name } { Nova~De~Hi }
\prop_put:Nnn \my_prop { gender } { male }
\prop_put:Nnn \my_prop { age } { 15 }

\prop_get:NnN \my_prop { name } \l_nova_name_tl
\prop_get:NnN \my_prop { gender } \l_nova_gender_tl
\prop_get:NnN \my_prop { age } \l_nova_age_tl

\begin{tabular}{|l|l|l|}
\hline
 Name & Gender & Age \\ \hline
```

## 10 Environment

```
\newenvironment{name}{beginning}{ending}
\renewenvironment{name}{beginning}{ending}
```

## 11 list environment

```
\newenvironment{mylist}{%
    \begin{list}{\mysmiley}{\itshape}
}{%
    \end{list}
}
```

☺ *Wonder Girls*

☺ *Girls' Generation*

### 11.1 Exercise

```
\usepackage{xstring}

\newcommand\selectedmark{\mysym{2611}}
```

```
 \l_nova_name_tl & \l_nova_gender_tl & \l_nova_age_tl \\
\hline
\end{tabular}
```

| Name | Gender | Age |
|---|---|---|
| Nova De Hi | male | 15 |

## 9 \NewDocumentEnvironment

```
\NewDocumentEnvironment { name } { args }
{ beginning }
{ ending }
```

\ProvideDocumentEnvironment, \RenewDocumentEnvironment가 더 있다.

## 10 Exercise

왼쪽의 Exercise 샘플을 재구현한 것이다. 원래 이 예제는 내가 쓴 일련의 글 선택지에 대하여, 선택지에 대하여 2와 선택지에 대하여 3에서 구현했던 코드이다.
이 코드의 발상(알고리즘)은 인자로 들어온 쉼표로 분리된 숫자들에 각각 + 기호를 붙여서 긴 텍스트열을 만들고 현재의 item counter가 이 텍스트열에 있는지(SubStr) 검사하여 만약 있다면 \selected를 마크하고 그렇지 않다면 \unselected를 마크하게 하는 것이었다.
expl3에서는 이렇게 할 필요가 없다. clist라는 쉼표로 분리된 자료 구조가 이미 있기 때문에 이를 이용하여 간단히 처리할 수 있다. 특히 \SubStr과 같은 명령을 쓸 필요 없이, expl3의 \clist_if_in:NnTF를 사용하였다. (여기서는 인자의 확장 문제로 \clist_if_in:NoTF 형식으로 쓰였다.) 즉, 인자로 들어온 쉼표로 분리된 숫자들을 \l_selnum_clist에 넣고, 현재 \item의 카운터에 대하여 \clist_if_in을 검사해서 이것이 참이면 \selected_box:를, 그렇지 않으면 \unselected_box:를 label로 찍는 list 환경을 만드는 것이다.

```
\newcommand\unselectedmark{\mysym{2610}}
\newcommand\selectednum{}
\newcounter{optionnum}
\newcommand\selectedlabel{
   \setcounter{optionnum}{0}
   \renewcommand\makelabel{
      \stepcounter{optionnum}
      \IfSubStr{\ch@ices}{+\theoptionnum+}%
         {\selectedmark}{\unselectedmark}}}
\def\ch@ices{}
\newenvironment{selected}[1]%
{\@for\@CurrentItem:=#1\do{%
 \edef\@CurrentItem{+\@CurrentItem+}
 \expandafter\edef\expandafter\ch@ices\expandafter%
   {\ch@ices\@CurrentItem}}
 \begin{list}{}{\selectedlabel}}%
{\end{list}}
```

```
My favorite idol girl groups are:
\begin{selected}{2,4,5}\tightlist
\item Wonder Girls
\item Girls' Generation
\item T-ara
\item Davichi
\item Secret
\item Sistar
\item 4minute
\end{selected}
```

My favorite idol girl groups are:

- ☐ Wonder Girls
- ☑ Girls' Generation
- ☐ T-ara
- ☑ Davichi

```
\int_new:N \g_item_counter_int

\cs_new_nopar:Nn \selected_box:
{
   \MySym { 2611 }
}

\cs_new_nopar:Nn \unselected_box:
{
   \MySym { 2610 }
}

\cs_new_nopar:Nn \selected_label:
{
   \int_zero:N \g_item_counter_int
   \cs_set:Nn \make_label_fn:
   {
      \int_gincr:N \g_item_counter_int
      \clist_if_in:NoTF \l_selnum_clist
             { \int_use:N \g_item_counter_int }
      { \selected_box: }
      { \unselected_box: }
   }

   \dim_set:Nn \labelwidth { 1em }
   \dim_set:Nn \labelsep   { 0.5em }
   \dim_set:Nn \leftmargin { 2.6em }
   \cs_set_eq:NN \makelabel \make_label_fn:
}

\NewDocumentEnvironment {Selected} { m }
{
   \clist_set:Nn \l_selnum_clist { #1 }
   \begin{list} {} { \selected_label: }
```

- ☑ Secret
- ☐ Sistar
- ☐ 4minute

## 12 Converting a Command to an Environment: environ package

```
\usepackage{environ}

\NewEnviron{Foo}[2][]{\foo[#1]{#2}{\BODY}}
```

```
}
{
   \end{list}
}
```

```
My favorite idol girl groups are:
\begin{Selected}{2,4,5}\tightlist
\item Wonder Girls
\item Girls' Generation
\item T-ara
\item Davichi
\item Secret
\item Sistar
\item 4minute
\end{Selected}
```

My favorite idol girl groups are:

- ☐ Wonder Girls
- ☑ Girls' Generation
- ☐ T-ara
- ☑ Davichi
- ☑ Secret
- ☐ Sistar
- ☐ 4minute

```
\begin{Foo}[akey=,...]
...
\end{Foo}
```

## 13 Miscellaneous Tips

### 13.1 \let, \relax

```
\let
\relax
```

```
\def\aaa{AAA}
\def\bbb{BBB}
\def\ccc{CCC}
\let\kkk\aaa
\kkk\par
\let\kkk\bbb
\kkk\par
\let\kkk\ccc
\kkk\par
\let\kkk\relax
\kkk\par
\ifx\kkk\undefined Undefind\else Defined\fi
```

AAA

BBB

CCC

Defined

### 13.2 case

```
\ifcase\value{cnt} \or ... \or ... \or\fi
```

## 12 Miscellaneous Tips

### 12.1 \tl_set_eq, \tl_clear

```
\cs_set_eq:NN <cs1> <cs2>
\tl_set_eq:NN <tl1> <tl2>
\tl_clear:N <tl>
```

```
\tl_set:Nn \aaa {AAA}
\tl_set:Nn \bbb {BBB}
\tl_set:Nn \ccc {CCC}
\tl_set_eq:NN \kkk \aaa
\kkk \par

\tl_set_eq:NN \kkk \bbb
\kkk \par

\tl_set_eq:NN \kkk \ccc
\kkk \par

\tl_clear:N \kkk
\kkk \par

\tl_if_empty:NTF \kkk { Empty } { Not Empty } ,~
\tl_if_blank:oTF \kkk { Blank } { Not Blank }
```

AAA

BBB

CCC

Empty, Blank

\ifcase 문은 정수 카운터를 기준으로 실행된다. \or은 이 카운터를 하나 증가시킨다. 예컨대 \dummycnt가 있다고 하고,

```
\ifcase\number\dummycnt
   % dummycnt=0일 때
\or
   % dummycnt=1일 때
\or
   % dummycnt=2일 때
...
\fi
```

이런 식이다.
일반적으로, \the\counter나 \number\counter보다 xifthen 패키지의 \value{cnt}를 쓰는 것이 오류 발생 가능성을 줄여준다. expl3의 case문은 이보다도 더 안정적이다. 다음 보기에서 MyCnt의 값이 4로 설정되어 있기 때문에 아래 \ifcase 문을 실행한 결과는 'Davichi'일 것이다.

```
\newcounter{MyCnt}
\setcounter{MyCnt}{4}
\ifcase\value{MyCnt}%
% 0
\or Wonder Girls      % 1
\or Girls' Generation % 2
\or T-ara             % 3
\or Davichi           % 4
\or Secret            % 5
\or Sistar            % 6
\or 4minute           % 7
\else                 % default
\fi
```

Davichi
스페이스를 무시하지 않는 일반 LaTeX 코딩의 특성상 \or를 각 항목 바로 뒤에 붙여 EOL 문자의 영향을 없애는 것이 관행이다. 즉,

```
\ifcase\value{MyCnt}\or
```

### 12.2 cases

```
\if_case:w \or: \fi:
\tl_case:Nn(TF)
\str_case:nn(TF)
\dim_case:nn(TF)
\int_case:nn(TF)
```

\if_case:w \or: \fi:는 \ifcase \or \fi와 동등하지만 expl3의 내부 명령이므로 되도록 사용하지 않는 것이 좋다. 그렇지만 사용례는 보이기로 한다.

```
\int_set:Nn \l_tmpa_int { 4 }
\if_case:w \l_tmpa_int
\or:
   Wonder~Girls
\or:
   Girls'~Generation
\or:
   T-ara
\or:
   Davichi
\or:
   Sistar
\else:
   Girl Groups
\fi:
```

Davichi
그러나, 같은 역할을 하는 \int_case:nnTF가 훨씬 읽기 쉽고 코딩상의 오류도 적게 발생한다.

```
\int_set:Nn \l_tmpa_int { 4 }
\int_case:nnTF { \int_use:N \l_tmpa_int }
{
    { 1 } { Wonder Girls }
    { 2 } { Girls' Generation }
    { 3 } { T-ara }
```

```
Wonder Girls\or
...
```

와 같이 입력하는 것이 통례이다.
이와 같이 소스 코드를 읽기 어렵게 하는 LaTeX 코딩 관행이 몇 가지 있다. expl3에서는 스페이스와 들여쓰기, 개행을 적극적으로 활용하여 소스 코드의 가독성을 높이려 하였다.

```
    { 4 } { Davichi }
    { 5 } { Secret }
    { 6 } { Sistar }
    { 7 } { 4minite }
}
{
    \space test~ succeeded!
}
{
    \space no~ matching~ girl~ group
}
```

Davichi test succeeded!
다음은 `\str_case` 문의 예이다. `\str_case_x:nnTF`는 `\str_case:nnTF`와 같지만 비교되는 스트링이 매크로일 때 그것을 확장해주는 역할을 한다. 여기서는 `\test_tl`이 매크로이므로 `\str_case_x`를 사용하였다.

```
\tl_set:Nn \test_tl { test }
\str_case_x:nnTF { \test_tl }
{
   { test } { TEST~ }
   { exam } { EXAM~ }
}
{
    test~ succeeded!
}
{
    No~matching~strings.
}
```

TEST test succeeded!

## 14 부록: 연산

### 14.1 정수 연산, counter

plain TEX

```
\newcount\MyCnt
\MyCnt=1
\advance\MyCnt by2  % 덧셈
\the\MyCnt
\advance\MyCnt by-1 % 뺄셈
\the\MyCnt
\multiply\MyCnt by2 % 곱셈
\the\MyCnt
\divide\MyCnt by3 % 버림 나눗셈
\the\MyCnt
```

3, 2, 4, 1.

$\varepsilon$-TEX

```
\MyCnt=1
\MyCnt=\numexpr (\MyCnt+2-1)*2/3\relax
\the\MyCnt
```

1

LATEX

```
\newcounter{cnt}
\setcounter{cnt}{1}
\stepcounter{cnt}
\addtocounter{cnt}{2}
\addtocounter{cnt}{-1}
\setcounter{cnt}{\numexpr \thecnt * 2 / 3\relax}
\thecnt
```

2

## 13 부록: 연산

### 13.1 정수 연산, expl3

```
\int_new:N \my_test_int
\int_zero:N \my_test_int
\int_set:Nn \my_test_int { 1 }

\int_incr:N \my_test_int   % 1 증가
\int_decr:N \my_test_int   % 1 감소
\int_add:Nn \my_test_int { 2 }  % 덧셈
\int_sub:Nn \my_test_int { 1 }  % 뺄셈

\int_set:Nn \my_test_int
   { \int_eval:n { ( \my_test_int * 3 + 10 ) / 2 } } % 일반 연산
\int_use:N \my_test_int ||

\int_add:Nn \my_test_int { 20 }

\int_set:Nn \my_test_int
   { \int_div_truncate:nn { \my_test_int } { 3 } } % 버림 나눗셈
\int_use:N \my_test_int ||
\int_set:Nn \my_test_int
   { \int_div_round:nn { \my_test_int } { 3 } } % 반올림 나눗셈
\int_use:N \my_test_int ||
\int_set:Nn \my_test_int
   { \int_mod:nn { \my_test_int } { 2 } } % 나머지 연산

\int_use:N \my_test_int
```

8||9||3||1

# 14 부록: 인자 지정자

## 14.1 xparse의 인자 지정자

**m** 일반 인자
**o** LaTeX의 표준 옵션 인자
**d** delimiter 지정 옵션 인자
**O** 기본값 지정 표준 옵션 인자
**D** 기본값 지정 delimiter 지정 옵션 인자
**s** star. `\IfBooleanTF`로 검사
**t** 기본값 지정 star형 인자. `\IfBooleanTF`로 검사.
**v** verbatim 옵션 인자. 다른 명령의 인자로 사용불가.

## 14.2 expl3의 인자 지정자

**n** 일반 토큰(열) 인자. {와 }로 둘러싼 표준 인자
**N** 토큰 인자 한 개 (단일 매크로)
**c** `\csname ...\endcsname`형 인자
**v** 변수 값 인자의 `\csname`형.
**V** 변수 값 인자 단일 매크로
**o** 한 번 확장 (expansion once)
**x** 가능한 한 확장 (exhaustive expansion)
**f** 첫 번째 매크로 확장 (full expansion)
**p** parameter
**w** weird. 부정형 인자
**D** Do not use. 내부함수이므로 일반적인 경우 사용 금지.