ksvqt — 「괄호(낫표)류」 입력

Nova de Hi

2025/07/20 ver 0.9.1

요 약

낫표류의 괄호형 부호 문자를 입력하는 방법을 정의한다. 사용자 부호를 지정하거나 부호 문자의 폭을 설정하는 방법도 함께 제공한다.

차 례

1 요약

우리 글 문장부호의 “낫표”와 “화살괄호”를 입력하는 것은 이들 부호가 키보드에 없기 때문에 약간 노력을 들여야 했다. 우리는 이 “낫표류” 부호를 간편하게 입력하는 방법을 제공하여 글쓰기의 생산성을 높이는 데 기여하고자 한다.

레거시 엔진은 제한적으로 지원한다. 3.5절에서 설명한다.

2 사용법

2.1 기본 명령

`\<...>`

이 패키지의 기본 명령이다. 텍스트 상황에서 이 명령은 다음과 같이 쓴다.

`\<0텍스트>`	「텍스트」

`\<` 다음에 괄호의 종류를 나타내는 숫자 인자 하나와 괄호로 감쌀 텍스트가 오고 이 명령을 종료하는 `>`를 끝에 둔다. `...`로 표시한 텍스트는 짧은 인자이다. 즉 `\par`를 포함하지 않는다.

괄호의 종류 숫자 인자로 나타내는 괄호의 종류는 다음과 같다.

0 「」	1 『』	2 〈〉	3 《》
4 〔〕	5 【】	6 (user, 〖〗)	7 (user, 〘〙)
8 (user, 〚〛)	9 (user, 〈〈〉〉)	! (default)	

0에서 5까지는 KS 기호문자이며 미리 정의되어 있다.

```
\<0텍스트>, \<1텍스트>, \<2텍스트>,
\<3텍스트>, \<4텍스트>, \<5텍스트>
```

「텍스트」,『텍스트』,〈텍스트〉,《텍스트》,〔텍스트〕,【텍스트】

6~9번은 사용자 정의를 위한 것이다. pdfLaTeX에서는 6번은 5번과 같은 검정대괄호, 7~9번은 0번과 같은 홑낫표로 찍힌다. 유니코드 엔진의 텍에서 6~9번은 각기 다음과 같은 괄호를 기본적으로 찍도록 되어 있기는 하나, 이 부호들은 모두 KS 기호문자가 아니므로 대부분의 한글 폰트로는 인쇄할 수 없을 것이다. 사용자는 6~9번을 자신이 원하는 부호로 정의하여 활용할 수 있는데 이에 대해서는 2.2절에서 다룬다.

```
\<6텍스트>, \<7텍스트>, \<8텍스트>, \<9텍스트>
```

〖텍스트〗, 〘텍스트〙, ⟦텍스트⟧, ⟪텍스트⟫

\ksvqtdefaultsym

특별한 부호로 !를 주는 것은 "기본 기호"를 강제하는 것이다. 만약 괄호 유형 지시자가 주어지지 않으면(즉 \< 직후 문자가 숫자가 아니면) 무조건 홑낫표가 사용된다. 이것을 "기본 기호"라고 한다.

```
\<텍스트> \<!텍스트>
```

「텍스트」「텍스트」

한 문서에서 한두 개의 괄호류 부호만 사용하게 되는 경우가 많을 것이다. 이럴 때 자신이 사용하려는 "기본 부호"의 번호를 미리 설정해둘 수 있다. 명령 \ksvqtdefaultsym은 숫자 인자를 하나 취하여 거기에 해당하는 괄호를 기본값으로 지정한다.

```
\ksvqtdefaultsym{5}
\<텍스트> \<!텍스트>
```

【텍스트】【텍스트】

2.2 사용자 부호 정의

6~9번에 대하여 사용자 부호를 (재)정의할 수 있다. 다음 \ksvqtuserdef 명령을 쓰는데 옵션 인자로는 정의할 부호의 번호를 6, 7, 8, 9 가운데서 지정한다.

\ksvqtuserdef

옵션 인자가 주어지지 않으면 7번을 설정한다. 필수 인자는 여는 부호와 닫는 부호 두 개로 이루어져 있어야 하고, 당연히 출력되는 폰트로 인쇄할 수 있어야 한다.

```
\<6텍스트> \<8텍스트> (변경전)

\ksvqtuserdef[6]{%
  {^^^^ff5b}{^^^^ff5d}}
\ksvqtuserdef[8]{{«}{»}}
\<6텍스트> \<8텍스트> (변경후)
```

〖텍스트〗⟦텍스트⟧ (변경전)
{텍스트} «텍스트» (변경후)

```
\ksvqtuserdef{{«}{»}}
\<7텍스트>
```

«텍스트»

유니코드 텍 엔진이라면 두 부호를 중괄호 없이 잇대어 정의하거나 유니코드 코드포인트를 부여하여도 동작한다. 그러나 pdfTEX 엔진에서는 이 두 부호 각각을 중괄호로 묶어주

어야 하고 유니코드 코드포인트로 지정할 수 없다(엄밀히 말하면 중괄호로 묶고 UTF-8 인코딩된 각 바이트를 지정하여야 한다).

```
\ksvqtuserdef[7]%
  {{^^^^ff5b}{^^^^ff5d}}
\<7텍스트>

\ksvqtuserdef[8]{‹›}
\<8텍스트>
```

{텍스트}
‹텍스트›

문서의 호환성을 고려하여 부호 각각을 중괄호로 묶는 방식을 권장한다.

\ksvqtuserdefrestore

(이 명령은 pdfLaTeX에서는 효력이 없다.) 옵션 인자로 6, 7, 8, 9 중에서 하나 이상을 (콤마로 구분하여) 지정할 수 있는데 주어진 번호에 해당하는 기호에 대해 사용자가 정의한 것을 취소하고 패키지의 초기값으로 되돌려 놓는다.

```
\ksvqtuserdef[8]{{«}{»}}
\<8텍스트>
\ksvqtuserdefrestore[8]
\<8텍스트>
```

«텍스트» ⟦텍스트⟧

\ksvqtspecmac

여는 부호와 닫는 부호를 인쇄하기 직전에 이 부호에만 영향을 주는 한 개의 사용자 매크로를 정의할 수 있다. 주로 현재 본문 서체로는 인쇄할 수 없는 폰트를 지정하기 위해서 마련한 명령이지만 \bfseries와 같은 일반적인 명령을 내릴 수도 있다. 명령은 옵션 인자로 6, 7, 8, 9 가운데 하나를 취할 수 있고 필수 인자를 해당 번호 문자의 매크로로 삼는다.

다음 예시에서 사용하고 있는 부호 [U+23A3]과 [U+23A6]은 현재 라틴 본문 서체인 TeX Gyre Pagella로는 인쇄할 수 없다.

```
\ksvqtuserdef[7]{{^^^^23a3}{^^^^23a6}}
\ksvqtspecmac[7]{\fontspec{FreeSans}}
\<7텍스트>
```

⎣텍스트⎦

특별히 6~9가 아닌 숫자를 옵션 인자에 넣으면 모든 기존 설정을 삭제한다. 옵션 인자를 주지 않으면 6번이다. 다음 예시는 폰트 설정이 주어진 번호에만 미치는 것, 이를 취소하는 것을 보인다.

```
\ksvqtuserdef[7]{{^^^^23a3}{^^^^23a6}}
\ksvqtspecmac[7]{\fontspec{FreeSerif}}
\<7텍스트> \<8텍스트>
\ksvqtspecmac[0]{}
\<7텍스트>
```

⌊텍스트⌋〖텍스트〗▫텍스트▫

2.3 부호 문자의 폭

`\ksvqtwidthdef`

유니코드 텍 엔진에서 괄호류 부호 문자는 폰트의 것을 프린트한다. 그런데 폰트마다 이 문자의 폭이 다르게 설정되어 있다. 그래서 우리는 괄호류(낫표류) 부호의 인쇄되는 폭(width)[1]을 강제하기로 하였고,[2] 기본값을 현재 사이즈의 0.5로 설정하였다.

`\ksvqtwidthdef` 명령은 1~7번 부호에 대하여 사용자가 폭을 재설정하게 한다.

```
\ksvqtwidthdef{<op_dim>}[<cl_dim>]
```

한 개의 필수 인자와 한 개의 옵션 인자를 줄 수 있고, 필수 인자만 있다면 여는 위치와 닫는 위치에 문자 폭을 동일하게 설정하며, 두 번째 옵션 인자는 닫는 위치 박스 폭을 여는 위치와 다르게 부여한다. 인자는 길이값(dimension) 형식이다.

이 설정은 지역적(*local*)이므로 현재의 group scope를 벗어나면 효력이 없다. 다음 보기는 눈에 띄게 하기 위해 일부러 큰 값을 설정한 것이다.

```
\ksvqtwidthdef{1em}
...\<0텍스트>...

\ksvqtwidthdef{1em}[2em]
...\<1텍스트>...
```

...「텍스트」...
...『텍스트』   ...

[1]이 매크로는 문자 박스의 width를 설정한다. `\hskip` 같은 것이 아니므로 예컨대 여는 부호가 행의 첫머리에 오더라도 박스의 폭은 설정한 대로 유지된다. 만약 `\hspace` 방식으로 간격을 설정하려 한다면 `\ksvqtoutermacro` 명령의 설명을 참조하라.

[2]따라서 luatexko를 쓰는 경우에도 `CompressPunctuations` 여부에 관계없이 일정한(원하는) 부호 문자의 폭을 유지하는 것이 가능하다.

\ksvqtwidthdef*

8번과 9번은 특별히 별표 붙인 명령으로 지정하여야 한다. 다음은 이 명령의 사용법이다.

```
\ksvqtwidthdef*[8]{<op_dim>}[<cl_dim>]
\ksvqtwidthdef*[9]{<op_dim>}[<cl_dim>]
```

첫 번째 옵션 인자는 8번인지 9번인지를 나타낸다. 생략하면 9번을 설정한다. 필수 인자는 마지막 옵션 인자가 없으면 여는 부호와 닫는 부호의 문자 폭을 설정하고 마지막 옵션 인자가 존재하면 여는 부호의 폭이 된다. 그 뒤의 옵션 인자는 닫는 부호의 문자 폭을 설정한다.

이 두 부호를 특별하게 취급하는 이유는 그 폭을 다른 것과 다르게 주어야 할 필요가 있을 때 대응하기 위해서이다. 애초값은 0.5em으로 되어 있고 지역적으로(*locally*) 설정하기 때문에 group scope를 벗어나면 효력이 없다.

예컨대 전각 기호문자를 괄호 문자 대용으로 쓰고자 할 때는 문자 폭이 1em 남짓이어야 할 것이다. 그런데 이것을 7번에 할당하고 부호문자 폭을 바꾸면 1~6번 괄호도 모두 폭이 달라진다. 그러므로 이와 같이 특별한 폭을 갖는 문자를 괄호류로 대용하려 한다면 8 또는 9번에 할당하는 것이 좋다.

```
\ksvqtuserdef[8]{{→}{←}}%
\ksvqtwidthdef*[8]{1.1em}
전각 \<8화살표>로 괄호대용
```

전각 →화살표← 로 괄호대용

다음 보기는 9번에 대하여 첫 옵션 인자로 [9]를 굳이 밝히지 않아도 되는 것, 그리고 group 범위를 벗어나면 원래 값이 복원되는 것을 시험한다. 눈에 띄게 하기 위하여 일부러 큰 값을 부여하였다.

```
\ksvqtuserdef[9]%
    {^^^^2018^^^^2019}
\begingroup
\ksvqtwidthdef*{2em}
abcd\<9작은따옴표>efgh
\endgroup\par
abcd\<9작은따옴표>efgh
```

abcd ‘작은따옴표’ efgh
abcd ‘작은따옴표’ efgh

\ksvqtwidthnatural
\ksvqtwidthforce

비록 부호 문자의 폭을 강제하는 것이 이 패키지의 설계 개념이기는 하지만 사용자는 이를 무력화하고 한글 식자 패키지의 설정을 따르게 할 수 있다. 명령이 주어지면 이 패키지의 문자 폭 강제가 억제된다. 이와 반대로 \ksvqtwidthforce는 이 패키지의 문자 폭을 강제한다. 이 두 명령은 옵션 인자 형식으로 [true], [false]를 줄 수도 있게 되어 있다.

다음 보기는 xetexko와 luatexko에서 다르게 나타날 것이다. 그래서 natural width를 선언한 뒤에는 \ksvqtwidthdef이 아무 효과가 없음을 보이는 방식으로 예를 들겠다.

```
\ksvqtwidthdef{2em}
\ksvqtwidthnatural
abcd\<!텍스트>xyz (natural)
\quad\ksvqtwidthforce
abcd\<!텍스트>xyz (default)
```

abcd「텍스트」xyz (natural) abcd 「텍스트」 xyz (default)

\ksvqtwidthnatural이 true 상태여도 다음 절에서 설명할 inner, outer 매크로는 동작하며, 0~9의 모든 부호에 대하여 유효하다.

\ksvqtwidthdefault

이 명령은 인자 없이, 모든 \ksvqtwidthdef에 의한 설정을 취소한다.

```
\ksvqtwidthdef{1.5em}
abcd\<!텍스트>xyz

\ksvqtwidthdefault
abcd\<!텍스트>xyz
```

abcd 「텍스트」 xyz
abcd「텍스트」xyz

\ksvqtcalcemsize
\ksvqtemsize

폰트 크기를 변경하는 명령(\small, \large, etc.)이 주어진 경우에 부호 문자의 폭이 고정되어 있어서 발생하는 문제를 피하기 위하여 \ksvqtemsize를 쓸 수 있다. 이 매크로가 의미하는 것은 다음과 같다. \ksvqtcalcemsize는 실수 인자를 취하여, 현재 폰트

크기에 곱한 값을 계산하여 \ksvqtemsize 매크로에 이를 저장한다. \ksvqtemsize는 길이 단위가 붙어 있지 않은 실수이나 폰트 크기를 point로 계산한 값이다.

```
\ksvqtcalcemsize{1}\ksvqtemsize \\
\ksvqtcalcemsize{0.5}\ksvqtemsize \\
\LARGE
\ksvqtcalcemsize{1} \ksvqtemsize
```

10.95

5.475

17.28

이 매크로를 활용하여 부호의 폭을 설정하려면 다음처럼 한다. 끝에 pt를 명시해야 한다는 점에 주의하라.

```
\LARGE
\ksvqtcalcemsize{.67}
\ksvqtwidthdef{\ksvqtemsize pt}
...\<1텍스트>...
```

...『텍스트』...

대체로 version 0.9 이후로는 \ksvqtwidthdef이 주어지지 않는 한 부호 문자의 width는 "현재 폰트 사이즈를 기준으로" 하고 있으므로 아주 특별한 경우가 아니면 이런 조작이 필요없을 것이다.

2.4 부호 문자의 inner, outer 매크로

\ksvqtinnermacro

여는 부호는 오른쪽 정렬, 닫는 부호는 왼쪽 정렬한다. 그리고 부호와 텍스트 사이에 원하는 매크로를 삽입할 수 있다. 명령의 사용 형식은 다음과 같다.

\ksvqtinnermacro[both|before|after]{<code>}

before는 여는 부호와 텍스트 사이, after는 텍스트와 닫는 부호 사이에 들어갈 매크로를 설정한다. both는 두 곳에 동일한 코드를 삽입한다. 옵션 인자를 주지 않으면 both이고, 들어가는 코드의 기본값은 아무 것도 하지 않는 것이다.

```
\ksvqtinnermacro[before]{\textcolor{red}{\rule{1pt}{10pt}}}
\ksvqtinnermacro*[after]{\textcolor{blue}{\rule{1pt}{10pt}}}
\<0텍스트>
\ksvqtinnermacro{\hspace{2pt}}
\<1텍스트>
```

「텍스트」『텍스트』

before나 after를 지정한 때에는 다른 쪽의 기존 설정은 지워지는데, 이전 설정을 지우지 않고 유지하게 하려면 별표를 붙인 \ksvqtinnermacro*를 쓴다.

이 설정값은 0~9의 모든 부호에 적용되고 효과는 지역적이다.

\ksvqtoutermacro

이 명령은 다음과 같이 사용한다.

```
\ksvqtoutermacro[both|before|after]{<code>}
```

before는 여는 부호를 찍기 직전에, after는 닫는 부호를 찍은 직후에 두는 매크로를 정의할 수 있고 both는 이 둘을 동일하게 설정한다. 옵션 인자를 주지 않으면 both이고, 기본값은 아무 것도 하지 않는 것이다.

```
\ksvqtoutermacro{\hspace{5pt}} %% both
abcd\<3텍스트>efgh,
\ksvqtoutermacro[before]{\textcolor{blue}{\rule{1em}{2pt}}}
\ksvqtoutermacro*[after]{\textcolor{red}{\rule{1em}{2pt}}}
abcd\<4텍스트>efgh
```

abcd《텍스트》efgh, abcd〔텍스트〕efgh

before나 after를 지정한 때에는 다른 쪽의 기존 설정은 지워지는데, 이전 설정값을 지우지 않고 유지되게 하려면 별표 붙인 명령 \ksvqtoutermacro*를 쓴다.

이 명령은 KTUG의 대화에서 "박스 폭을 변경하지 말고 간격을 주고 싶다"는 요청 때문에 마련한 것이었다. 이 설정값은 0~9의 모든 부호에 적용되고 효과는 지역적이다.

before 매크로가 선언형으로 주어졌을 때 선언의 효력은 >까지이다.

```
\ksvqtoutermacro[before]{\color{red}}
abcd\<3테스트>efgh
```

abcd《테스트》efgh

2.5 부호의 높이와 수직 이동

이 패키지는 부호의 vertical position과 height에 대해서는 아무런 변경도 가하지 않고 있으며 폰트 자체의 자면을 그대로 찍게 하고 있다. adjustbox를 이용하여 (필요하다면) 수직 위치와 높이를 제어하는 예를 들어 보겠다. 예컨대 〖텍스트〗와 같은 모양의 괄호 (`[U+27E6]`, `[U+27E7]`)를 사용하려 하는데(이 부호는 `[U+301A]`, `[U+301B]`와 닮았지만 모양이 조금 다르다), 크기를 조금 줄이고 수직 위치를 위로 올려 맞추어, 〖텍스트〗와 같은 모양을 만들려 하는 때를 생각한다. 폰트는 Noto Serif CJK KR이다.

```
\ksvqtuserdef[7]{%
  {\adjustbox{raise=.5ex,height=.7\height,width=\width}
   ↪  {\symbol{"27E6}}}%
  {\adjustbox{raise=.5ex,height=.7\height,width=\width}
   ↪  {\symbol{"27E7}}}%
}
\<7텍스트>

〖텍스트〗
```

이 코드가 참고가 되기를 바란다.

조금 머리를 쓰면 앞서 소개한 `\ksvqtspecmac`을 이용하여 다음처럼 할 수도 있을지 모른다.

```
\def\myspecmac#1{%
    \adjustbox{raise=.5ex,height=.7\height,%
               width=\width}{#1}}
\ksvqtuserdef[7]{{^^^^27e6}{^^^^27e7}}
\ksvqtspecmac[7]{\myspecmac}
\<7텍스트>

〖텍스트〗
```

3 주의 및 참고사항

3.1 명령어 형식

`\ksvqtBrackets`

만약 `\<`를 부득이하게 사용할 수 없는 상황에서도 이 패키지가 제공하는 입력 방식을 활용하려 한다면 `\ksvqtBrackets` 명령어로 대신할 수 있다. 이것은 `\<`와 완전히 동일하

므로 예컨대 다음과 같이 사용하여야 한다. 첫 토큰이 <가 아니면 아무런 일도 일어나지 않는다.

```
\ksvqtBrackets<1텍스트>
```

『텍스트』

3.2 tabbing 호환성

\>와 \<는 *tabbing* 환경에서 특별한 의미를 갖기 때문에 이 환경 내부에서는 원래의 의미로 쓰이게 하였다. 그러므로 *tabbing* 내부에서는 \<를 괄호류(낫표류)를 식자하는 데 사용할 수 없다. oblivoir의 저자는 *tabbing*을 되도록 쓰지 말라고 하고 있는 까닭에 이 문제에 대한 대응책은 마련하지 않지만 앞서 소개한 \ksvqtBrackets를 이용할 수는 있을 것이다.

3.3 숫자 인자

명령의 첫 부호가 괄호 유형을 나타내는 숫자이므로 식자하고자 하는 텍스트가 숫자로 시작할 때 주의해야 한다. 예를 들어 「2025」를 식자하고자 할 때 \<2025>로 하면 첫 숫자 2가 괄호 유형 인자로 인식되기 때문에 의도치 않은 결과(〈025〉)가 발생한다. \<{0}2025> 또는 \<!2025>와 같이 입력하여야 「2025」로 나타난다.

3.4 인자 안의 > 부호

예를 들어, 다음과 같은 경우를 생각하자.

```
\<!$a>b$>
```

이 패키지의 괄호 명령 \<와 \ksvqtBrackets는 그 끝위치를 > 부호로 찾으려 한다. 그런데 부등호나 혹은 다른 맥락에서 >가 필요해져서 이것을 인자 안에 넣으려 할 때는 괄호 명령의 끝나는 위치를 잘못 파악하게 될 가능성이 있다. 입력할 때 이런 문자가 포함되어 있으면 중괄호로 글마디를 묶어주어야 한다.

```
\<!{$a>b$}>
\<1가 {>} 나>
```

「$a > b$」『가 > 나』

3.5 pdfTEX 호환성

pdfTEX은 유니코드 엔진이 아니기 때문에 주의해야 할 점이 있다.

(1) 미리 정의된 괄호 종류 가운데 6번(〖 〗)은 나타나지 않는다. 6번을 지정하면 5번(【 】)과 같은 모양이다.

(2) 7~9번 사용자 괄호류를 정의할 때 반드시 두 부호를 각각 중괄호로 묶어주어야 한다.

(3) 7~9번 사용자 괄호류는 KS X 1001의 범위 안의 부호만 유효하다. 즉 유니코드 괄호류는 정의하더라도 나타나지 않는다.

4 변경 이력

2025/07/20. version 0.9.1 bug fix concerning linebreak
2025/07/03. version 0.9 `\ksvqtemsize`
2025/06/10. version 0.8 default+6~9. `\ksvqtspecmac`
2025/06/05. version 0.7 `\ksvqtwidthnatural`
2025/05/20. version 0.6 userdef+6, `\ksvqtBrackets`
2025/05/18. version 0.5 inner and outer macros
2025/05/17. version 0.4 `\ksvqtwidthdef`
2025/05/14. version 0.1

명령어 색인