esgutil package

이엑스피엘쓰리 스타디 그룹

2017년 7월 13일, version 0.0

차 례

1 popping from token list 1
2 string compare 2
3 int step break 3
4 fp format 3
5 split by plus sign 4

1 popping from token list

```
\esg_tl_pop:NN <tokenlist1> <tokenlist2>
\esg_tl_pop:nN { <tokenlist1> } <tokenlist2>
\esg_tl_pop:nNN { <tokenlist> } <tokenlist1> <tokenlist2>
```

<tokenlist1>의 첫 번째 item을 <tokenlist2>에 넣고 <tokenlist1>에서 삭제한다.
:nN은 <tokenlist1>의 첫 번째 아이템을 <tokenlist2>에 넣고 첫 번째 아이템이 삭제된 tl을 입력 스트림에 남긴다.
:nNN은 첫 번째 토큰리스트를 head와 tail로 나누어 <tokenlist1>과 <tokenlist2>에 넣는다. 입력 스트림에 아무 것도 남기지 않는다.

```
\ExplSyntaxOn
\tl_set:Nn \l_tmpa_tl { abcdefg }
\l_tmpa_tl \par

\esg_tl_pop:NN \l_tmpa_tl \l_tmpb_tl
\l_tmpb_tl;~\l_tmpa_tl \par

\esg_tl_pop:NN \l_tmpa_tl \l_tmpb_tl
\l_tmpb_tl;~\l_tmpa_tl \par

\ExplSyntaxOff
```

abcdefg
a; bcdefg
b; cdefg

```
\ExplSyntaxOn
\esg_tl_pop:nN { abcdefg } \l_tmpa_tl

\par \l_tmpa_tl \par
\ExplSyntaxOff
```

bcdefg
a

```
\ExplSyntaxOn
\esg_tl_pop:nNN { abcdefg } \l_tmpa_tl
    \l_tmpb_tl
<\l_tmpa_tl>; <<\l_tmpb_tl>>
\ExplSyntaxOff
```

<a>;«bcdefg»

2 string compare

```
\esg_str_cmp:nn { <string1> } { <string2> }
\esg_str_cmd:Vn <string1> { <string2> }
```

\pdftex_strcmp:D의 범용 버전. LuaTEX에서도 동작한다.

```
\ExplSyntaxOn
\clist_set:Nn \l_tmpa_clist
{
    Kodiak, cheetah, Puma, Jaguar,
    Panther, Tiger, Leopard,
    Snow~Leopard, Lion, Mountain~Lion,
    mavericks, Yosemite, El~Capitan,
    Sierra, High~Sierra, Mojave,
    Catalina
}

\clist_sort:Nn \l_tmpa_clist
{
    \int_compare:nTF { \esg_str_cmp:nn
    { \str_fold_case:n { #1 } } {
    \str_fold_case:n { #2 } } > 0 }
    { \sort_return_swapped: } {
    \sort_return_same: }
}

\clist_use:Nn \l_tmpa_clist {\\}

\ExplSyntaxOff
```

Catalina
cheetah
El Capitan
High Sierra
Jaguar
Kodiak
Leopard
Lion
mavericks
Mojave
Mountain Lion
Panther
Puma
Sierra
Snow Leopard
Tiger
Yosemite

3 int step break

```
\esg_int_step_break:
```

`\int_step_inline:` 또는 `\int_step_function:`을 중단하는 명령

```
\ExplSyntaxOn
\cs_new:Npn \s_func:n #1
{
    \int_compare:nT { #1 >= 15 } { \esg_int_step_break: }
    \textit{#1} |~
}

\int_step_function:nnnN { 2 } { 2 } { 20 } \s_func:n
\ExplSyntaxOff
```

2| *4*| *6*| *8*| *10*| *12*| *14*|

4 fp format

```
\esgfpformat[<option>]{number} (Document Command)
\esg_fp_format:nn <integer> <fpexpress>
```

문서명령 `\esgfpformat`의 옵션 인자 `<option>`에는 숫자가 온다. round할 위치. default=0.

```
\esgfpformat[int]{fp_expr}
\esg_fp_format:nn <int> <fp_expr>
```

이 둘은 (거의) 동일하고 문서 명령이 robust라는 것만 다르다.
`\esg_fp_format:nn`의 첫 인자는 소수 몇 째자리까지 보일 것인지를 나타내는 정수이다. 이 숫자가 0이면 round한 정수를 보여준다. n이 음수이면 10^n 자리에서 반올림한 결과를 보여준다. 양수이면 소숫점 아래 n째 자리(10^{-n})에서 반올림한 결과를 보여준다.
두 번째 n인자는 fp expression이다. `\fp_eval:n`이 적용된다.
입력 스트림에 남겨지는 것은 number (int or fp)가 아니라 문자열(tl)이다.

```
\esgfpformat[4]{102} \\
\ExplSyntaxOn
\esg_fp_format:nn { 4 } { sqrt ( 2 ) }
\ExplSyntaxOff
```

102.0000
1.4142

VV, nV, no variant가 있다. 매크로로 묶인 변수를 인자로 줄 때에 사용한다.

```
\ExplSyntaxOn
\fp_set:Nn \l_tmpa_fp { sqrt ( pi ) }
\int_set:Nn \l_tmpa_int { 3 }
```

```
\esg_fp_format:VV \l_tmpa_int \l_tmpa_fp
\ExplSyntaxOff
```

```
1.772
```

5 split by plus sign

```
\esg_split_plussign:nNN
```

#1로 주어지는 인자의 형태를 분석하여 + 부호를 전후로 분리하고 앞 부분을 #2에, 뒷 부분을 #3에 넣어 반환한다. 반환값은 모두 tl이다. #1에 + 부호가 들어 있지 않다면 에러. +가 둘 이상 들어 있으면 첫 번째 것만을 기준으로 분리한다.
variant로 :VNN이 있다.

```
\ExplSyntaxOn
\esg_split_plussign:nNN { 1+2+3 } \l_xx_tl \l_yy_tl
\tl_use:N \l_xx_tl \par
\tl_use:N \l_yy_tl \par
\esg_split_plussign:VNN \l_yy_tl \l_xx_tl \l_zz_tl
\tl_use:N \l_xx_tl \par
\tl_use:N \l_zz_tl
\ExplSyntaxOff
```

```
1
2+3
2
3
```