The **argproc** package

Handle user input into integer, dimen, skip, and muskip expressions

Oliver Beery

Version 0.0.0   5 February 2025

1 Loading the package

```
% LaTeX2e version 2023-11-01 added \IfExplAtLeastTF.
\NeedsTeXFormat{LaTeX2e}[2023-11-01]
\ProvidesExplPackage
  {argproc}
  {2025-02-05}
  {0.0.0}
  {Handle user input into integer, dimen, skip, and muskip expressions.}

% l3kernel version 2023-10-10 added many 'e'-variants.
\IfExplAtLeastTF { 2023-10-10 } { }
  {
    \msg_new:nnn { argproc } { expl3-out-of-date }
      {
        The~ argproc~ package~ could~ not~ load.~
        This~ package~ requires~
        L3~ programming~ layer~ version~ 2023-10-10~ or~ newer.
      }
    \msg_critical:nn { argproc } { expl3-out-of-date }
  }
```

2 Messages

```
\msg_new:nnn { argproc } { int-expr-invalid }
  { Invalid~ integer~ expression~ '#1'~ \msg_line_context:. }
\msg_new:nnn { argproc } { dim-expr-invalid }
  { Invalid~ dimen~ expression~ '#1'~ \msg_line_context:. }
\msg_new:nnn { argproc } { skip-expr-invalid }
  { Invalid~ skip~ expression~ '#1'~ \msg_line_context:. }
\msg_new:nnn { argproc } { muskip-expr-invalid }
  { Invalid~ muskip~ expression~ '#1'~ \msg_line_context:. }
```

3 Variables

```
% Stores user input into the expression as a string. This is used in the error
% messages.
\str_new:N \l__argproc_int_expr_str
```

```
\str_new:N \l__argproc_dim_expr_str
\str_new:N \l__argproc_skip_expr_str
\str_new:N \l__argproc_muskip_expr_str

% Stores the appended default unit.
\tl_new:N \l__argproc_dim_expr_unit_tl
\tl_new:N \l__argproc_skip_expr_unit_tl

% Scratch variables
\int_new:N    \l__argproc_tmp_int
\dim_new:N    \l__argproc_tmp_dim
\skip_new:N   \l__argproc_tmp_skip
\muskip_new:N \l__argproc_tmp_muskip
```

4 Primitives

```
\cs_new_eq:NN \__argproc_assignment_insert_after:N \afterassignment

\cs_new_eq:NN \__argproc_int_expr:w    \numexpr
\cs_new_eq:NN \__argproc_dim_expr:w    \dimexpr
\cs_new_eq:NN \__argproc_skip_expr:w   \glueexpr
\cs_new_eq:NN \__argproc_muskip_expr:w \muexpr

\cs_new_eq:NN \__argproc_int_expr_end:    \relax
\cs_new_eq:NN \__argproc_dim_expr_end:    \relax
\cs_new_eq:NN \__argproc_skip_expr_end:   \relax
\cs_new_eq:NN \__argproc_muskip_expr_end: \relax
```

5 Setting integers from user input

```
\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2
      {
        \str_set:Nn \l__argproc_int_expr_str {##2}
        \__argproc_assignment_insert_after:N \__argproc_int_chk:w
          #2 \__argproc_int_expr:w ##2 \__argproc_int_expr_end:
        \__argproc_int_chk_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_int_set_from_user:Nn  { #1 = }
\__argproc_tmp:Nn \argproc_int_gset_from_user:Nn { \global #1 = }
\__argproc_tmp:Nn \argproc_int_add_from_user:Nn  { \advance #1 }
```

```
\__argproc_tmp:Nn \argproc_int_gadd_from_user:Nn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_int_sub_from_user:Nn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_int_gsub_from_user:Nn { \global \advance #1 - }

% Issues an error if the user inputs any extra trailing tokens that were not
% processed by the integer expression.
\cs_new_protected:Npn \__argproc_int_chk:w #1 \__argproc_int_chk_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_int_expr_end: }
      {
        \msg_error:nnV { argproc } { int-expr-invalid }
          \l__argproc_int_expr_str
      }
  }
\cs_new_eq:NN \__argproc_int_chk_end: \scan_stop:
```

6 Setting dimens from user input

```
% \scan_stop: is used here for the case where a skip variable is incorrectly
% provided as #1.
\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2##3
      {
        \str_set:Nn \l__argproc_dim_expr_str {##2}
        \tl_set:Ne \l__argproc_dim_expr_unit_tl {##3}
        \__argproc_assignment_insert_after:N \__argproc_dim_chk:w
          #2 \__argproc_dim_expr:w ##2 \l__argproc_dim_expr_unit_tl
            \__argproc_dim_expr_end: \scan_stop:
        \__argproc_dim_chk_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_dim_set_from_user:Nnn  { #1 = }
\__argproc_tmp:Nn \argproc_dim_gset_from_user:Nnn { \global #1 = }
\__argproc_tmp:Nn \argproc_dim_add_from_user:Nnn  { \advance #1 }
\__argproc_tmp:Nn \argproc_dim_gadd_from_user:Nnn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_dim_sub_from_user:Nnn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_dim_gsub_from_user:Nnn { \global \advance #1 - }

% When testing whether the user has added extra trailing tokens to the dimen
% expression, there are 3 cases:
% (1) No extra tokens were appended. \__argproc_dim_expr_end: was gobbled by
% the dimen expression. \l__argproc_dim_expr_unit_tl was either used in the
% assignment or was blank.
% (2) Extra tokens were appended before expanding \l__argproc_dim_expr_unit_tl,
% e.g. \argproc_dim_set_from_user:Nnn \l_tmpa_dim { 10pt<tokens> } { <unit> }
% (3) If \l__argproc_dim_expr_unit_tl was expanded, it should either be used in
% the assignment (case 1) or left in place---we need to test that the default
% unit was not half-gobbled, e.g. \argproc_dim_set_from_user:Nnn \l_tmpa_dim
% { 10b } { pt }, resulting in \l_tmpa_dim=10bp where 't' is the extra token.
\cs_new_protected:Npn \__argproc_dim_chk:w #1 \__argproc_dim_chk_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_dim_expr_end: }
      {
        \tl_if_in:nnTF {#1} { \l__argproc_dim_expr_unit_tl }
          {
            \msg_error:nnV { argproc } { dim-expr-invalid }
              \l__argproc_dim_expr_str
          }
          {
            \str_if_eq:noF {#1}
              {
                \l__argproc_dim_expr_unit_tl \__argproc_dim_expr_end:
                  \scan_stop:
              }
              {
                \msg_error:nnV { argproc } { dim-expr-invalid }
                  \l__argproc_dim_expr_str
              }
          }
      }
  }
\cs_new_eq:NN \__argproc_dim_chk_end: \scan_stop:

\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2
      {
        \str_set:Nn \l__argproc_dim_expr_str {##2}
        \__argproc_assignment_insert_after:N \__argproc_dim_chk_no_unit:w
          #2 \__argproc_dim_expr:w ##2 \__argproc_dim_expr_end: \scan_stop:
        \__argproc_dim_chk_no_unit_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_dim_set_from_user:Nn  { #1 = }
\__argproc_tmp:Nn \argproc_dim_gset_from_user:Nn { \global #1 = }
\__argproc_tmp:Nn \argproc_dim_add_from_user:Nn  { \advance #1 }
\__argproc_tmp:Nn \argproc_dim_gadd_from_user:Nn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_dim_sub_from_user:Nn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_dim_gsub_from_user:Nn { \global \advance #1 - }

\cs_new_protected:Npn \__argproc_dim_chk_no_unit:w #1
```

```
  \__argproc_dim_chk_no_unit_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_dim_expr_end: }
      {
        \msg_error:nnV { argproc } { dim-expr-invalid }
          \l__argproc_dim_expr_str
      }
  }
\cs_new_eq:NN \__argproc_dim_chk_no_unit_end: \scan_stop:
```

7 Setting skips from user input

```
\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2##3
      {
        \str_set:Nn \l__argproc_skip_expr_str {##2}
        \tl_set:Ne \l__argproc_skip_expr_unit_tl {##3}
        \__argproc_assignment_insert_after:N \__argproc_skip_chk:w
          #2 \__argproc_skip_expr:w ##2 \l__argproc_skip_expr_unit_tl
            \__argproc_skip_expr_end:
        \__argproc_skip_chk_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_skip_set_from_user:Nnn  { #1 = }
\__argproc_tmp:Nn \argproc_skip_gset_from_user:Nnn { \global #1 = }
\__argproc_tmp:Nn \argproc_skip_add_from_user:Nnn  { \advance #1 }
\__argproc_tmp:Nn \argproc_skip_gadd_from_user:Nnn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_skip_sub_from_user:Nnn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_skip_gsub_from_user:Nnn { \global \advance #1 - }

% See the comments for \__argproc_dim_chk:w.
\cs_new_protected:Npn \__argproc_skip_chk:w #1 \__argproc_skip_chk_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_skip_expr_end: }
      {
        \tl_if_in:nnTF {#1} { \l__argproc_skip_expr_unit_tl }
          {
            \msg_error:nnV { argproc } { skip-expr-invalid }
              \l__argproc_skip_expr_str
          }
          {
            \str_if_eq:noF {#1}
              { \l__argproc_skip_expr_unit_tl \__argproc_skip_expr_end: }
              {
                \msg_error:nnV { argproc } { skip-expr-invalid }
                \l__argproc_skip_expr_str
            }
        }
    }
  }
\cs_new_eq:NN \__argproc_skip_chk_end: \scan_stop:

\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2
      {
        \str_set:Nn \l__argproc_skip_expr_str {##2}
        \__argproc_assignment_insert_after:N \__argproc_skip_chk_no_unit:w
          #2 \__argproc_skip_expr:w ##2 \__argproc_skip_expr_end:
        \__argproc_skip_chk_no_unit_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_skip_set_from_user:Nn  { #1 = }
\__argproc_tmp:Nn \argproc_skip_gset_from_user:Nn { \global #1 = }
\__argproc_tmp:Nn \argproc_skip_add_from_user:Nn  { \advance #1 }
\__argproc_tmp:Nn \argproc_skip_gadd_from_user:Nn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_skip_sub_from_user:Nn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_skip_gsub_from_user:Nn { \global \advance #1 - }

\cs_new_protected:Npn \__argproc_skip_chk_no_unit:w #1
  \__argproc_skip_chk_no_unit_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_skip_expr_end: }
      {
        \msg_error:nnV { argproc } { skip-expr-invalid }
          \l__argproc_skip_expr_str
      }
  }
\cs_new_eq:NN \__argproc_skip_chk_no_unit_end: \scan_stop:
```

8 Setting muskips from user input

```
\cs_set_protected:Npn \__argproc_tmp:Nn #1#2
  {
    \cs_new_protected:Npn #1 ##1##2
      {
        \str_set:Nn \l__argproc_muskip_expr_str {##2}
        \__argproc_assignment_insert_after:N \__argproc_muskip_chk:w
          #2 \__argproc_muskip_expr:w ##2 \__argproc_muskip_expr_end:
        \__argproc_muskip_chk_end:
      }
    \cs_generate_variant:Nn #1 { NV , c , cV }
  }
\__argproc_tmp:Nn \argproc_muskip_set_from_user:Nn  { #1 = }
\__argproc_tmp:Nn \argproc_muskip_gset_from_user:Nn { \global #1 = }
\__argproc_tmp:Nn \argproc_muskip_add_from_user:Nn  { \advance #1 }
\__argproc_tmp:Nn \argproc_muskip_gadd_from_user:Nn { \global \advance #1 }
\__argproc_tmp:Nn \argproc_muskip_sub_from_user:Nn  { \advance #1 - }
\__argproc_tmp:Nn \argproc_muskip_gsub_from_user:Nn { \global \advance #1 - }

% Issues an error if the user inputs any extra trailing tokens that were not
% processed by the muskip expression.
\cs_new_protected:Npn \__argproc_muskip_chk:w #1 \__argproc_muskip_chk_end:
  {
    \tl_if_in:nnT {#1} { \__argproc_muskip_expr_end: }
      {
        \msg_error:nnV { argproc } { muskip-expr-invalid }
          \l__argproc_muskip_expr_str
      }
  }
\cs_new_eq:NN \__argproc_muskip_chk_end: \scan_stop:
```

9 Argument processors

```
\cs_new_protected:Npn \argproc_int:n #1
  {
    \argproc_int_set_from_user:Nn \l__argproc_tmp_int {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_int
  }
\cs_new_protected:Npn \argproc_dim:nn #1#2
  {
    \argproc_dim_set_from_user:Nnn \l__argproc_tmp_dim {#2} {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_dim
  }
\cs_new_protected:Npn \argproc_dim:n #1
  {
    \argproc_dim_set_from_user:Nn \l__argproc_tmp_dim {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_dim
  }
\cs_new_protected:Npn \argproc_skip:nn #1#2
  {
    \argproc_skip_set_from_user:Nnn \l__argproc_tmp_skip {#2} {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_skip
  }
\cs_new_protected:Npn \argproc_skip:n #1
  {
    \argproc_skip_set_from_user:Nn \l__argproc_tmp_skip {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_skip
  }
\cs_new_protected:Npn \argproc_muskip:n #1
  {
    \argproc_muskip_set_from_user:Nn \l__argproc_tmp_muskip {#1}
    \tl_set:NV \ProcessedArgument \l__argproc_tmp_muskip
  }
```