ltx-talk – A class for typesetting presentations*

Joseph Wright†

Released 2025-12-01

Contents

I ltx-talk – Overall set up 1

1 ltx-talk implementation 1
1.1 Set up . . . 1
1.2 Additions for expl3 . . . 1
1.3 Extra variants . . . 2
1.4 Scratch space . . . 2
1.5 Option handling . . . 2
1.6 Setting up . . . 3
1.7 Math support . . . 4
1.8 Font selection . . . 4
1.9 Hyperlinks . . . 5
1.10 Tagging . . . 5

II ltx-talk-color – Color definitions 6

1 ltx-talk-color implementation 6
1.1 Existing definitions . . . 6
1.2 Document commands . . . 6
1.3 Color definition . . . 7
1.4 Semantic colors . . . 8

III ltx-talk-decode – Decoding overlay specs 9

1 ltx-talk-decode implementation 9

IV ltx-talk-frame – The structure of frames 16

*This file describes v0.3.4, last revised 2025-12-01.

†E-mail: joseph@texdev.net

1 ltx-talk-frame implementation 16
1.1 Slides in frames . . . 16
1.2 Counters . . . 19
1.3 Frame options . . . 20
1.4 Tagging for headers . . . 20
1.5 Wallpaper . . . 21
1.6 The `frame` environment . . . 25

V ltx-talk-frame – The structure of frames 28

1 ltx-talk-frame-structure implementation 28
1.1 Columns . . . 28
1.2 Floats . . . 30
1.3 Footnotes . . . 32

VI ltx-talk-mode – Modes 33

1 ltx-talk-mode implementation 33

VII ltx-talk-overlay – Overlays 34

1 ltx-talk-overlay implementation 34
1.1 Utilities . . . 34
1.2 Action commands and environments . . . 34
1.3 Non-action commands and environments . . . 38
1.4 Fixed-size areas . . . 39
1.5 Adding overlays to existing commands . . . 41

VIII ltx-talk-required – "Required" definitions 44

1 ltx-talk-required implementation 44
1.1 Standard design settings . . . 44
1.2 List support . . . 45

IX ltx-talk-structure – Structural commands 46

1 ltx-talk-structure implementation 46
1.1 Frame title . . . 46
1.2 Sectioning . . . 47
1.3 Table of contents . . . 49
1.4 Block environments . . . 51
1.5 Lists . . . 52
1.6 Theorems, *etc.* . . . 55

X ltx-talk-title – Title pages 56

1 ltx-talk-title implementation 56

Index 60

Part I
ltx-talk – Overall set up

1 ltx-talk implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

1.1 Set up

Identify the package and give the over all version information.

```
3 \ProvidesExplClass {ltx-talk} {2025-12-01} {0.3.4}
4   {A class for typesetting presentations}
```

Get the right type of message.

```
5 \prop_gput:Nnn \g_msg_module_name_prop { talk } { ltx-talk }
6 \prop_gput:Nnn \g_msg_module_type_prop { talk } { Class }
```

Require the latest LaTeX structures.

```
7 \IfFormatAtLeastF{2025-11-01}
8   {
9     \msg_new:nnnn { ltx-talk } { kernel-too-old }
10      { The~ltx-talk~class~requires~LaTeX~2025-11-01~or~later. }
11      {
12        You~have~tried~to~use~the~ltx-talk~class~with~a~LaTeX~kernel~release~
13        prior~to~2025-11-01;~the~required~functionality~is~missing.
14      }
15    \msg_fatal:nn { ltx-talk } { kernel-too-old }
16  }
17 \NeedsDocumentMetadata
```

1.2 Additions for expl3

Like `\vcoffin_set:Nnn`, so should be an easy enough addition.

```
18 \cs_gset_protected:Npn \vbox_set_to_wd:Nnn #1#2#3
19   {
20     \tex_setbox:D #1 \tex_vbox:D
21       {
22         \tex_hsize:D \__box_dim_eval:n {#2}
23         \color_group_begin: #3 \par \color_group_end:
24       }
25     \box_dp:N #1 \__box_dim_eval:n {#2}
26   }
27 \cs_gset_protected:Npn \vbox_set_to_wd:Nnw #1#2
28   {
29     \cs_set_protected:Npn \__box_set_to_wd:
30       { \box_wd:N #1 \__box_dim_eval:n {#2} }
31     \tex_setbox:D #1 \tex_vbox:D
32       \c_group_begin_token
```

```
        \tex_hsize:D \__box_dim_eval:n {#2}
        \group_insert_after:N \__box_set_to_wd:
        \color_group_begin:
  }
```

Some things from xbox that would be useful.

```
\cs_gset_protected:Npn \rule:nnn #1#2#3
  {
    \tex_vrule:D
      height \dim_eval:n {#2} \exp_stop_f:
      depth  \dim_eval:n {#3} \exp_stop_f:
      width  \dim_eval:n {#1} \exp_stop_f:
      \scan_stop:
  }
```

1.3 Extra variants

```
\cs_generate_variant:Nn \clist_set:Nn { cv }
\cs_generate_variant:Nn \hook_gput_code:nnn { nne }
\exp_args_generate:n { nVv }
\cs_generate_variant:Nn \color_select:n { V }
\cs_generate_variant:Nn \dim_compare:nNnTF { v }
\cs_generate_variant:Nn \dim_compare_p:nNn { vNv }
\cs_generate_variant:Nn \dim_max:nn { v }
\cs_generate_variant:Nn \str_replace_all:Nnn { NnV }
\cs_generate_variant:Nn \text_purify:n { v }
\cs_generate_variant:Nn \vbox_to_ht:nn { v }
```

1.4 Scratch space

__talk_tmp:w

For one-off processing.

```
\cs_new_protected:Npn \__talk_tmp:w { }
```

(*End of definition for* `\__talk_tmp:w`.)

\l__talk_tmp_box

```
\box_new:N \l__talk_tmp_box
```

(*End of definition for* `\l__talk_tmp_box`.)

\l__talk_tmp_tl

```
\tl_new:N \l__talk_tmp_tl
```

(*End of definition for* `\l__talk_tmp_tl`.)

1.5 Option handling

\l__talk_aspect_ratio_str
\l__talk_fontsize_dim
\l__talk_frame_title_bool
\l__talk_mode_str

```
\keys_define:nn { talk }
  {
    aspect-ratio .str_set:N =
      \l__talk_aspect_ratio_str ,
    font-size .dim_set:N =
      \l__talk_fontsize_dim ,
    frame-title-arg .bool_set:N =
```

```
      \l__talk_frame_title_bool ,
    handout .code:n =
      { \str_set:Nn \l__talk_mode_str { handout } } ,
    handout .value_forbidden:n = true ,
    mode .choices:nn =
      { handout , projector }
      { \str_set:NV \l__talk_mode_str \l_keys_choice_tl }
  }
```

(*End of definition for* `\l__talk_aspect_ratio_str` *and others.*)

Scope for options.

```
\keys_define:nn { talk }
  {
    aspect-ratio    .usage:n = load ,
    font-size       .usage:n = load ,
    frame-title-arg .usage:n = load ,
    mode            .usage:n = load
  }
```

Initial values.

```
\keys_set:nn { talk }
  {
    aspect-ratio    = 16:9      ,
    font-size       = 11pt      ,
    frame-title-arg = false     ,
    mode            = projector
  }
\ProcessKeyOptions [ talk ]
```

1.6 Setting up

Load the font size setup if available, otherwise fall back on scaling.

```
\file_if_exist_input:nF { size \dim_to_decimal:n \l__talk_fontsize_dim .clo }
  {
    \file_input:n { size10.clo }
    \RequirePackage { relsize }
    \hook_gput_code:nne { begindocument } { talk }
      { \exp_not:N \relsize { \fp_eval:n { \l__talk_fontsize_dim / 10pt } } }
  }
```

`\c__talk_paper_height_dim`
`\c__talk_paper_width_dim`

As geometry is being used to set the paper size with no previous value, we have to use the optional argument rather than waiting to apply `\geometry`.

```
\dim_const:Nn \c__talk_paper_height_dim { 100mm }
\use:e
  {
    \cs_set_protected:Npn \exp_not:N \__talk_tmp:w
      #1 \tl_to_str:n { : } #2 \tl_to_str:n { : } #3 \exp_not:N \q_stop
      {
        \dim_const:Nn \exp_not:N \c__talk_paper_width_dim
          {
            \exp_not:N \fp_to_dim:n
              { (#1 / #2) * \exp_not:N \c__talk_paper_height_dim }
          }
```

```
      }
    \exp_not:N \__talk_tmp:w \l__talk_aspect_ratio_str
      \tl_to_str:n { : } 100 \exp_not:N \q_stop
  }
\use:e
  {
    \exp_not:N \RequirePackage
      [
        papersize =
          {
            \dim_use:N \c__talk_paper_width_dim  ,
            \dim_use:N \c__talk_paper_height_dim
          } ,
        tmargin    = 10mm ,
        bmargin    =  8mm ,
        lmargin    = 10mm ,
        rmargin    = 10mm ,
        headheight = 10mm ,
        headsep    =  2mm ,
        footskip   =  6mm
      ]
      { geometry }
  }
```

(*End of definition for* `\c__talk_paper_height_dim` *and* `\c__talk_paper_width_dim`.)

Turn off justification

```
\raggedright
```

1.7 Math support

We always require amsmath: this is forced anyway by unicode-math for LuaTEX.

```
\RequirePackage { amsmath }
```

1.8 Font selection

The aim here is to minimize change from the standard font setup but at the same time provide a sans-serif default. Since beamer was released, better sans-serif math mode fonts have become available. For OpenType engines, requiring unicode-math is the most sensible approach; we also load mathtools as that has to be before unicode-math. The New Computer Modern font provides a reasonable initial set of glyphs. It comes with a wrapper package, but that does various other things: if the user wants these, they can choose to load themselves. For 8-bit engines, switching the text font to be sans-serif is easy. For math mode, the sansmathfonts package does a good job: here, using the package rather than adjusting directly is the sensible option.

```
\sys_if_engine_opentype:TF
  {
    \RequirePackage { mathtools }
    \RequirePackage { unicode-math }
    \setsansfont { NewCMSans10-Regular.otf }
    \setmathfont { NewCMSansMath-Regular.otf }
  }
  {
```

```
    \RequirePackage { sansmathfonts }
    \RequirePackage [ nomath ] { lmodern }
  }
\cs_set_eq:NN \rmdefault \sfdefault
```

To ensure that math mode fonts are always initialized, force loading at the start of the document. This is left as late as possible: just before typesetting starts. This is needed to set up math dimensions for vertical centering.

```
\AddToHook { begindocument / end } { \check@mathfonts }
```

1.9 Hyperlinks

\thepage We define \thepage here: this is checked for by hyperref so has to come early.

```
\cs_new:Npn \thepage { \@arabic \c@page }
```

(*End of definition for* \thepage*. This variable is documented on page* ??.)

A requirement.

```
\RequirePackage { hyperref }
\hypersetup { hidelinks }
```

1.10 Tagging

We need to extend the standard tagging model to work with slides and so on.

```
\tagpdfsetup
  {
    role / user-NS = ltx-talk        ,
    role / new-tag = frame / Sect    ,
    role / new-tag = frametitle / H4
  }
```

⟨/class⟩

Part II
ltx-talk-color – Color definitions

1 ltx-talk-color implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

The aim here is to *test* how well l3color can support the range of color functions that are needed for a presentation. As such, this is very much experimental, but deliberately so. In particular, there is an important question about the need for global colors: used throughout beamer but otherwise not widely encountered. At the same time, there is a need to work with packages that expect color to be managed in a predictable way: pgf in particular makes use of xcolor internal as part of color management.

Currently, colors defined using xcolor will be passed on to l3color provided `\DocumentMetadata` is active. As that is a requirement in any case for ltx-talk, some of the setup is relatively easy to do.

1.1 Existing definitions

```
3 \RequirePackage { xcolor }
```

`\stdcolor` `\stdmathcolor` `\stdtextcolor` Save the document commands.

```
4 \NewCommandCopy \stdcolor \color
5 \NewCommandCopy \stdmathcolor \mathcolor
6 \NewCommandCopy \stdtextcolor \textcolor
```

(*End of definition for* `\stdcolor`, `\stdmathcolor`, *and* `\stdtextcolor`. *These functions are documented on page* ??.)

1.2 Document commands

```
7 \cs_generate_variant:Nn \color_select:n { e }
8 \cs_generate_variant:Nn \color_select:nn { ne }
9 \cs_generate_variant:Nn \color_math:nn { e }
10 \cs_generate_variant:Nn \color_math:nnn { ne }
```

`\color` `\mathcolor` `\textcolor` Add the overlay specification and use l3color.

```
11 \RenewDocumentCommand \color { D <> { all } o m }
12   {
13     \__talk_if_overlay:nT {#1}
14       {
15         \IfNoValueTF {#2}
16           { \color_select:e {#3} }
17           { \color_select:ne {#2} {#3} }
18       }
19     \ignorespaces
20   }
21 \RenewDocumentCommand \mathcolor { D <> { all } o m +m }
22   {
```

```
    \__talk_if_overlay:nT {#1}
      {
        \IfNoValueTF {#2}
          { \color_math:en {#3} {#4} }
          { \color_math:nen {#2} {#3} {#4} }
      }
  }
\RenewDocumentCommand \textcolor { D <> { all } o m +m }
  {
    \__talk_if_overlay:nT {#1}
      {
        \mode_leave_vertical:
        \group_begin:
          \IfNoValueTF {#2}
            { \color_select:e {#3} }
            { \color_select:ne {#2} {#3} }
          #4
        \group_end:
      }
  }
```

(*End of definition for* `\color`*,* `\mathcolor`*, and* `\textcolor`*. These functions are documented on page* **??**.)

`\pagecolor` Here, the definition is different: we directly use the shipout hook.

`\__talk_pagecolor:n`

```
\RenewDocumentCommand \pagecolor { D <> { all } o m }
  {
    \__talk_if_overlay:nT {#1}
      {
        \IfNoValueTF {#2}
          { \__talk_pagecolor:n { {#3} } }
          { \__talk_pagecolor:n { [ {#2} ] {#3} } }
      }
  }
\cs_new_protected:Npn \__talk_pagecolor:n #1
  {
    \AddToHook { shipout / background }
      {
        \color #1
        \put ( 0cm, -\paperheight )
          { \rule { \paperwidth } { \paperheight } }
      }
  }
```

(*End of definition for* `\pagecolor` *and* `\__talk_pagecolor:n`*. This function is documented on page* **??**.)

1.3 Color definition

`\DeclareColor` Provide a single interface here: as the data will be passed to l3color in any case, there is not too much to do.

```
\NewDocumentCommand \DeclareColor { m o m }
  {
    \IfNoValueTF {#2}
      { \colorlet {#1} {#3} }
```

```
      { \definecolor {#1} {#2} {#3} }
  }
```

(*End of definition for* `\DeclareColor`*. This function is documented on page* **??**.)

1.4 Semantic colors

Pick up the standard colors from beamer.

```
\DeclareColor { alert } [ RGB ] { 200 , 0 , 0 }
\DeclareColor { example } { green!50!black }
\DeclareColor { structure } [ rgb ] { 0.2 , 0.2 , 0.7 }
```

⟨/class⟩

Part III
ltx-talk-decode – Decoding overlay specs

1 ltx-talk-decode implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

\l__talk_decode_overlays_bool

The result from decoding: are we on the current slide. This may well be better handled by moving to a `TF` signature: to be explored.

```
3 \bool_new:N \l__talk_decode_overlays_bool
```

(*End of definition for* `\l__talk_decode_overlays_bool`.)

\g__talk_pauses_int
\c@pauses
\thepauses

The automatically-incremented value for the relative overlay value.

```
4 \int_new:N \g__talk_pauses_int
5 \cs_new_eq:NN \c@pauses \g__talk_pauses_int
6 \cs_new:Npn \thepauses { \@arabic \g__talk_pauses_int }
```

(*End of definition for* `\g__talk_pauses_int`, `\c@pauses`, *and* `\thepauses`. *These variables are documented on page* ??.)

\l__talk_decode_pure_bool

Tracks whether only mode specifications were given.

```
7 \bool_new:N \l__talk_decode_pure_bool
```

(*End of definition for* `\l__talk_decode_pure_bool`.)

\l__talk_decode_step_bool

Tracks whether to step `\g__talk_pauses_int`.

```
8 \bool_new:N \l__talk_decode_step_bool
```

(*End of definition for* `\l__talk_decode_step_bool`.)

\l__talk_decode_arg_str

For error usage.

```
9 \str_new:N \l__talk_decode_arg_str
```

(*End of definition for* `\l__talk_decode_arg_str`.)

\l__talk_decode_overlays_clist
\l__talk_decode_overlays_str

The decoded overlay specification: will have only absolute slide numbers present, potentially along with ranges.

```
10 \clist_new:N \l__talk_decode_overlays_clist
11 \str_new:N \l__talk_decode_overlays_str
```

(*End of definition for* `\l__talk_decode_overlays_clist` *and* `\l__talk_decode_overlays_str`.)

\l__talk_decode_action_str

The action which is active, if any.

```
12 \str_new:N \l__talk_decode_action_str
```

(*End of definition for* `\l__talk_decode_action_str`.)

\l__talk_decode_actions_bool
\l__talk_decode_actions_clist
\l__talk_decode_actions_str

For the actions versions of overlay tracking.

```
\bool_new:N \l__talk_decode_actions_bool
\clist_new:N \l__talk_decode_actions_clist
\str_new:N \l__talk_decode_actions_str
```

(*End of definition for* `\l__talk_decode_actions_bool`, `\l__talk_decode_actions_clist`, *and* `\l__talk_decode_actions_str`.)

__talk_decode_parse:n
__talk_decode_parse_aux:n
__talk_decode_parse:w

First a simple check for an entirely blank argument: if that's the case, there is no additional overlay to consider. Then deal with any category code issues before looping over blocks divided by | tokens.

```
\cs_new_protected:Npn \__talk_decode_parse:n #1
  {
    \str_clear:N \l__talk_decode_action_str
    \bool_lazy_or:nnTF
      { \tl_if_blank_p:n {#1} }
      { \str_if_eq_p:nn {#1} { all } }
      { \bool_set_true:N \l__talk_decode_overlays_bool }
      {
        \str_set:Nn \l__talk_decode_arg_str {#1}
        \bool_set_false:N \l__talk_decode_actions_bool
        \bool_set_false:N \l__talk_decode_overlays_bool
        \bool_set_true:N \l__talk_decode_pure_bool
        \str_clear:N \l__talk_decode_overlays_str
        \str_clear:N \l__talk_decode_actions_str
        \exp_args:No \__talk_decode_parse_aux:n { \l__talk_decode_arg_str }
      }
  }
\cs_new_protected:Npn \__talk_decode_parse_aux:n #1
  { \__talk_decode_parse:w #1 | \q_recursion_tail | \q_recursion_stop }
```

The end-of-loop test here covers the case where the active mode is not mentioned at all in the specification.

```
\cs_new_protected:Npn \__talk_decode_parse:w #1 |
  {
    \quark_if_recursion_tail_stop_do:nn {#1}
      {
        \bool_lazy_and:nnT
          { \str_if_empty_p:N \l__talk_decode_overlays_str }
          { ! \l__talk_decode_pure_bool }
          { \bool_set_true:N \l__talk_decode_overlays_bool }
      }
    \exp_args:Ne \__talk_decode_mode:n
      { \tl_trim_spaces:n {#1} }
    \__talk_decode_parse:w
  }
```

(*End of definition for* `\__talk_decode_parse:n`, `\__talk_decode_parse_aux:n`, *and* `\__talk_decode_parse:w`.)

\c__talk_modes_clist

The possible modes: detokenized as that is applied up-front in decoding.

```
\clist_const:Ne \c__talk_modes_clist
  {
    \tl_to_str:n { handout }   ,
    \tl_to_str:n { projector }
  }
```

(*End of definition for* `\c__talk_modes_clist`.)

`\__talk_decode_mode:n` `\__talk_decode_mode:w` `\__talk_decode_mode_aux:n`

Check if the mode is known and current. If we find an action but have no overlay details, they are filled in with a `*`.

```
\cs_new_protected:Npe \__talk_decode_mode:n #1
  {
    \clist_if_in:NnTF \exp_not:N \c__talk_modes_clist {#1}
      {
        \exp_not:N \str_if_eq:VnT
          \exp_not:N \l__talk_mode_str {#1}
          { \bool_set_true:N \exp_not:N \l__talk_decode_overlays_bool }
      }
      {
        \exp_not:N \__talk_decode_mode:w #1 \tl_to_str:n { : : }
          \exp_not:N \q_stop
      }
  }
\use:e
  {
    \cs_new_protected:Npe \exp_not:N \__talk_decode_mode:w
      #1 \token_to_str:N :
      #2 \token_to_str:N :
      #3 \exp_not:N \q_stop
  }
  {
    \exp_not:N \tl_if_blank:nTF {#2}
      {
        \exp_not:N \__talk_decode_mode:nn
          { \tl_to_str:n { projector } } {#1}
      }
      { \exp_not:N \__talk_decode_mode:nn {#1} {#2} }
  }
\cs_new_protected:Npn \__talk_decode_mode:nn #1#2
  {
    \str_if_eq:VnTF \l__talk_mode_str {#1}
      {
        \__talk_decode_action:n {#2}
        \str_if_empty:NT \l__talk_decode_overlays_str
          { \__talk_decode_overlays:nn { overlays } { * } }
      }
      {
        \tl_if_blank:nF {#2}
          { \bool_set_false:N \l__talk_decode_pure_bool }
      }
  }
```

(*End of definition for* `\__talk_decode_mode:n`, `\__talk_decode_mode:w`, *and* `\__talk_decode_mode_aux:n`.)

`\__talk_decode_action:n` `\__talk_decode_action:w`

Here, we have two valid possibilities: no action specification at all, or from the known list. If we don't find one of those outcomes, we can issue an error.

```
\cs_new_protected:Npe \__talk_decode_action:n #1
  {
    \exp_not:N \__talk_decode_action:w
```

```
      #1 \tl_to_str:n { @ @ } \exp_not:N \q_stop
  }
\use:e
  {
    \cs_new_protected:Npn \exp_not:N \__talk_decode_action:w
      #1 \tl_to_str:n { @ } #2 \tl_to_str:n { @ } #3 \exp_not:N \q_stop
  }
  {
    \tl_if_blank:nTF {#2}
      { \__talk_decode_overlays:nn { overlays } {#1} }
      {
        \cs_if_exist:cTF { __talk_action_ #1 :N }
          {
            \bool_set_false:N \l__talk_decode_pure_bool
            \str_set:Nn \l__talk_decode_action_str {#1}
            \tl_if_blank:nF {#2}
              { \__talk_decode_overlays:nn { actions } {#2} }
          }
          {
            \msg_error:nnV { talk } { bad-action-spec }
              \l__talk_decode_arg_str
           }
      }
  }
```

(*End of definition for* `\__talk_decode_action:n` *and* `\__talk_decode_action:w`.)

`\__talk_decode_overlays:nn`
`\__talk_decode_overlays:nN`
`\@_decode_overlay_+:nw`
`\__talk_decode_overlay_.:nw`
`\__talk_decode_overlay_aux:nNN`
`\__talk_decode_overlay_offset:nNnN`
`\__talk_decode_overlay_offset:nNn`

The loop here needs to replace all `+` and `.` characters by the current automatic value, allowing for any offsets. This step also needs to track whether to increment the automatic value: true if a `+` is seen, false otherwise.

```
\cs_new_protected:Npn \__talk_decode_overlays:nn #1#2
  {
    \bool_set_false:N \l__talk_decode_step_bool
    \__talk_decode_overlays:nN {#1} #2 \q_recursion_tail \q_recursion_stop
    \bool_if:NT \l__talk_decode_step_bool
      { \int_gincr:N \g__talk_pauses_int }
    \__talk_decode_check:n {#1}
  }
\cs_new_protected:Npn \__talk_decode_overlays:nN #1#2
  {
    \quark_if_recursion_tail_stop:N #2
    \cs_if_exist_use:cF { __talk_decode_overlay_ #2 :nw }
      {
        \str_put_right:cn { l__talk_decode_ #1 _str } {#2}
        \__talk_decode_overlays:nN
      }
        {#1}
  }
\cs_new_protected:cpn { __talk_decode_overlay_+:nw } #1
  {
    \bool_set_true:N \l__talk_decode_step_bool
    \__talk_decode_overlay_aux:nNN {#1} 1
  }
\cs_new_protected:cpn { __talk_decode_overlay_.:nw } #1
```

```
  { \__talk_decode_overlay_aux:nNN {#1} 0 }
```

The look-ahead for an offset to a relative specification. If the end-of-loop is reached, the value still needs to be inserted: to share auxiliaries, that is done by using the same function as elsewhere, so the end-of-loop markers are re-inserted. Otherwise, there is a check to see if the next token is a `(`.

```
\cs_new_protected:Npn \__talk_decode_overlay_aux:nNN #1#2#3
  {
    \quark_if_recursion_tail_stop_do:Nn #3
      {
        \__talk_decode_overlay_offset:nNn {#1} #2 { 0 }
        \q_recursion_tail \q_recursion_stop
      }
    \token_if_eq_meaning:NNTF #3 ( % )
      { \__talk_decode_overlay_offset:nNnN {#1} #2 { } }
      { \__talk_decode_overlay_offset:nNn {#1} #2 { 0 } #3 }
  }
```

For the end of an offset, any valid overlay specification must have a closing `)`, so this time the end-of-loop case is an error. Otherwise simply collect up tokens until the closing `)` is found.

```
\cs_new_protected:Npn \__talk_decode_overlay_offset:nNnN #1#2#3#4
  {
    \quark_if_recursion_tail_stop_do:Nn #4
      {
        \msg_error:nnV { talk } { bad-action-spec }
          \l__talk_decode_arg_str
      } % (
    \token_if_eq_meaning:NNTF #4 )
      { \__talk_decode_overlay_offset:nNn {#1} #2 {#3} }
      { \__talk_decode_overlay_offset:nNnN {#1} #2 {#3#4} }
  }
```

Overlay values can never be negative: this is enforced here.

```
\cs_new_protected:Npn \__talk_decode_overlay_offset:nNn #1#2#3
  {
    \str_put_right:ce { l__talk_decode_ #1 _str }
      { \int_max:nn { 0 } { #3 + \g__talk_pauses_int + #2 } }
    \__talk_decode_overlays:nN {#1}
  }
```

(*End of definition for* `\__talk_decode_overlays:nn` *and others. This function is documented on page* **??**.)

`\__talk_decode_check:n`
`\__talk_decode_check:nw`
`\__talk_decode_check_single:nn`
`\__talk_decode_check_range:nnn`

At this stage we have a fully "written out" overlay specification, and need to work out if the current slide is included. We need to look at each entry in the comma-separated list to sort this out. First we filter out the case of a `*`, then it's a question of working out whether each entry is a single number or a range, and if the latter, whether it's open at either the start or the end.

```
\cs_new_protected:Npn \__talk_decode_check:n #1
  {
    \clist_set:cv { l__talk_decode_ #1 _clist } { l__talk_decode_ #1 _str }
    \clist_if_in:cnTF { l__talk_decode_ #1 _clist } { * }
      { \bool_set_true:c { l__talk_decode_ #1 _bool } }
      {
```

```
        \clist_map_inline:cn { l__talk_decode_ #1 _clist }
          { \__talk_decode_check:nw {#1} 0 ##1 - - \q_stop }
      }
  }
```

If `#4` is empty, both of the "filler" `-` tokens were consumed: we have a single value. Otherwise there is a range: the setup above ensures that there will be a starting value in all cases due to the leading `0`, but there may not be an end one.

```
\cs_new_protected:Npn \__talk_decode_check:nw #1#2 - #3 - #4 \q_stop
  {
    \tl_if_empty:nTF {#4}
      { \__talk_decode_check_single:nn {#1} {#2} }
      {
        \tl_if_blank:nTF {#3}
          { \__talk_decode_check_range:nnn {#1} {#2} { \c_max_int } }
          { \__talk_decode_check_range:nnn {#1} {#2} {#3} }
      }
  }
\cs_set_protected:Npn \__talk_decode_check_single:nn #1#2
  {
    \int_compare:nNnTF \g__talk_slide_int = {#2}
      {
        \bool_set_true:c { l__talk_decode_ #1 _bool }
        \clist_map_break:
      }
      {
        \int_compare:nNnT {#2} > \g__talk_slide_int
          { \bool_gset_true:N \g__talk_slide_continue_bool }
      }
  }
```

TODO: In the following we might want to add a check whether the range was given with `#2` being smaller than `#3`, to be decided upon.

```
\cs_set_protected:Npn \__talk_decode_check_range:nnn #1#2#3
  {
    \int_compare:nNnF \g__talk_slide_int > {#3}
      {
        \int_compare:nNnTF \g__talk_slide_int < {#2}
          { \bool_gset_true:N \g__talk_slide_continue_bool }
          {
            \bool_set_true:c { l__talk_decode_ #1 _bool }
            \bool_lazy_and:nnT
              { \int_compare_p:nNn \g__talk_slide_int < {#3} }
              { \int_compare_p:nNn {#3} < \c_max_int }
              { \bool_gset_true:N \g__talk_slide_continue_bool }
            \clist_map_break:
          }
      }
  }
```

(*End of definition for* `\__talk_decode_check:n` *and others.*)

```
\msg_new:nnnn { talk } { bad-action-spec }
  { Bad~overlay~specification~"#1". }
  {
    The~overlay~specification~given~doesn't~follow~the~pattern~described~in~
```

```
    the~ltx-talk~manual:~it~has~been~ignored.
  }
```
⟨/class⟩

Part IV
ltx-talk-frame – The structure of frames

1 ltx-talk-frame implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

1.1 Slides in frames

Currently, each slide in a frame will produce a separate page in the output (unless the slide is suppressed entirely). Material is then hidden on some pages by using opacity. An alternative approach would be to use Optional Content Groups to have a similar effect on one page per frame. However, whilst that would be relatively clear for appear/disappear effects, it would be much less straight-forward for partial transparency, *etc.*, plus would depend more heavily on viewer support. At a future stage we may wish to revisit this.

`\g__talk_slide_continue_bool` Tracks whether the frame continues after the current slide.

```
3 \bool_new:N \g__talk_slide_continue_bool
```

(*End of definition for* `\g__talk_slide_continue_bool`.)

`\l__talk_slide_box`

```
4 \box_new:N \l__talk_slide_box
```

(*End of definition for* `\l__talk_slide_box`.)

`\g__talk_slide_int` `\c@slide` `\theslide` The slide number inside the current frame: needed to know which overlays are active. We also provide LaTeX counter-style access.

```
5 \int_new:N \g__talk_slide_int
6 \cs_new_eq:NN \c@slide \g__talk_slide_int
7 \cs_new:Npn \theslide { \@arabic \c@slide }
```

(*End of definition for* `\g__talk_slide_int`, `\c@slide`, *and* `\theslide`. *These variables are documented on page* ??.)

Required to know which is the last slide in a frame for tagging.

```
8 \property_new:nnnn { slides } { now } { 1 } { \int_use:N \g__talk_slide_int }
```

`\__talk_slide:nn` `\__talk_slide_aux:n` Each slide is parsed inside simple set up, the only complexity being if we are handling fragile frames. There, all `\obeyedline` in the grabbed tokens need to be turned back into `^^M` before rescanning: this ensures that any verbatim grabbing in the frame still works. The strange business with setting the continuation boolean is needed as otherwise we get an infinite loop if there is an overlay specification for the frame itself. Tagging should not of itself force slide continuation, so the global boolean is reset for the tagged slide.

```
9 \cs_new_protected:Npn \__talk_slide:nn #1#2
10  {
```

```
    \group_begin:
      \tl_set:Ne \l__talk_tmp_tl
        {
          \property_ref:ee { frame . \int_use:N \g__talk_frame_int }
            { slides }
        }
      \str_if_eq:VnTF \l__talk_frame_tagging_str { n }
        { \str_set:NV \l__talk_frame_tagging_str \l__talk_tmp_tl }
        {
          \str_replace_all:NnV \l__talk_frame_tagging_str { ,n }
            \l__talk_tmp_tl
          \str_replace_all:NnV \l__talk_frame_tagging_str { ,~n }
            \l__talk_tmp_tl
        }
      \int_gzero:N \g__talk_slide_int
      \RenewCommandCopy \frame \__talk_latexe_frame:n
      \bool_do_while:Nn \g__talk_slide_continue_bool
        {
          \int_gincr:N \g__talk_slide_int
          \bool_gset_false:N \g__talk_slide_continue_bool
          \__talk_if_overlay:nT {#1}
            {
              \__talk_slide_begin:
              \__talk_if_overlay:VTF \l__talk_frame_tagging_str
                {
                  \bool_gset_false:N \g__talk_slide_continue_bool
                  \__talk_frame_tag:n
                }
                {
                  \bool_gset_false:N \g__talk_slide_continue_bool
                  \__talk_frame_notag:n
                }
                  {
                    \bool_if:NTF \l__talk_frame_verb_bool
                      { \__talk_slide_aux:n }
                      { \use:n }
                        {#2}
                  }
               \__talk_slide_end:
            }
        }
      \property_record:ee { frame . \int_use:N \g__talk_frame_int }
        { slides }
    \group_end:
  }
\cs_new_protected:Npn \__talk_slide_aux:n #1
  {
    \group_begin:
      \cs_set:Npn \obeyedline { ^^J }
      \use:e
        {
          \group_end:
          \tl_retokenize:n {#1}
        }
```

```
  }
```

(*End of definition for* `\__talk_slide:nn` *and* `\__talk_slide_aux:n`.)

The very last frame will not be recorded by the above, so we have to add to the hook at the very end of the run.

```
\AddToHook { enddocument / afterlastpage }
  {
    \property_record:ee { frame . \int_use:N \g__talk_frame_int }
      { slides }
  }
```

`\g__talk_frame_struct_int` The tagging structure number for the slide: needed by the content placed outside of the current box (for example the frame title).

```
\int_new:N \g__talk_frame_struct_int
```

(*End of definition for* `\g__talk_frame_struct_int`.)

`\__talk_slide_begin:`
`\__talk_slide_end:`

```
\cs_new_protected:Npn \__talk_slide_begin:
  {
    \int_gzero:N \g__talk_pauses_int
    \tl_gclear:N \g__talk_frame_title_tl
    \tl_gclear:N \g__talk_frame_subtitle_tl
    \__talk_cnt_save:
    \vbox_set:Nw \l__talk_slide_box
      \tl_gclear:N \g__talk_onslide_tl
  }
\cs_new_protected:Npn \__talk_slide_end:
  {
      \tl_use:N \g__talk_onslide_tl
    \vbox_set_end:
    \bool_if:NT \g__talk_slide_continue_bool
      { \__talk_cnt_restore: }
    \vbox_to_ht:nn { \textheight }
      {
        \use:c { __talk_slide_align_ \l__talk_frame_alignment_tl :n }
          { \vbox_unpack_drop:N \l__talk_slide_box }
      }
    \clearpage
  }
```

(*End of definition for* `\__talk_slide_begin:` *and* `\__talk_slide_end:`.)

`\__talk_slide_align_bottom:n` A pretty standard abstraction: we make sure there are always two skips.
`\__talk_slide_align_center:n`
`\__talk_slide_align_stretch:n`
`\__talk_slide_align_top:n`

```
\cs_new_protected:Npn \__talk_slide_align_bottom:n #1
  {
    \skip_vertical:n { 0pt~plus~1fil }
    #1
    \skip_vertical:n { 0pt }
  }
\cs_new_protected:Npn \__talk_slide_align_center:n #1
  {
    \skip_vertical:n { 0pt~plus~0.5fil }
    #1
```

```
    \skip_vertical:n { 0pt~plus~0.5fil }
  }
\cs_new_protected:Npn \__talk_slide_align_stretch:n #1
  {
    \skip_vertical:n { 0pt }
    #1
    \skip_vertical:n { 0pt }
  }
\cs_new_protected:Npn \__talk_slide_align_top:n #1
  {
    \skip_vertical:n { 0pt }
    #1
    \skip_vertical:n { 0pt~plus~1fil }
  }
```

(*End of definition for* `\__talk_slide_align_bottom:n` *and others.*)

1.2 Counters

`\l__talk_cnt_reset_seq`

As `\stepcounter`, *etc.*, will increment at each overlay, there is a need to save and reset. The list will be finalized at the end of the preamble, so the data storage is created at that point. The starting point is counters created before the class is loaded (other than those for lists, which reset "naturally"). Other cases are handled by adding to `\newcounter`.

```
\seq_new:N \l__talk_cnt_reset_seq
\seq_set_from_clist:Nn \l__talk_cnt_reset_seq
  {
    equation       ,
    footnote       ,
    mpfootnote     ,
    parentequation
  }
\seq_map_inline:Nn \l__talk_cnt_reset_seq
    {
      \int_new:c { g__talk_saved_ #1 _int }
      \int_gset_eq:cc { g__talk_saved_ #1 _int } { c@ #1 }
    }
```

(*End of definition for* `\l__talk_cnt_reset_seq`.)

`\__talk_cnt_save:`
`\__talk_cnt_restore:`

A simple save-and-restore pair.

```
\cs_new_protected:Npn \__talk_cnt_save:
  {
    \seq_map_inline:Nn \l__talk_cnt_reset_seq
      { \int_gset_eq:cc { g__talk_saved_ ##1 _int } { c@ ##1 } }
  }
\cs_new_protected:Npn \__talk_cnt_restore:
  {
    \seq_map_inline:Nn \l__talk_cnt_reset_seq
      { \int_gset_eq:cc { c@ ##1 } { g__talk_saved_ ##1 _int } }
  }
```

(*End of definition for* `\__talk_cnt_save:` *and* `\__talk_cnt_restore:`.)

\@definecounter
\std@definecounter

Track all counters for resetting.

```
\cs_new_eq:NN \std@definecounter \@definecounter
\cs_gset_protected:Npn \@definecounter #1
  {
    \std@definecounter {#1}
    \int_new:c { g__talk_saved_ #1 _int }
    \seq_gput_right:Nn \l__talk_cnt_reset_seq {#1}
  }
```

(*End of definition for* `\@definecounter` *and* `\std@definecounter`*. These functions are documented on page* ??.)

1.3 Frame options

\l__talk_frame_alignment_tl

```
\tl_new:N \l__talk_frame_alignment_tl
```

(*End of definition for* `\l__talk_frame_alignment_tl`.)

\l__talk_action_spec_str
\l__talk_frame_tagging_str

```
\keys_define:nn { talk / frame }
  {
    action-spec .str_set:N
      = \l__talk_action_spec_str ,
    tag-slides .str_set:N
      = \l__talk_frame_tagging_str ,
    vertical-alignment .choices:nn =
      { bottom , center , stretch , top }
      {
        \tl_set_eq:NN \l__talk_frame_alignment_tl
          \l_keys_value_tl
      }
  }
\keys_set:nn { talk / frame }
  {
    action-spec        =        ,
    tag-slides         = n      ,
    vertical-alignment = center
  }
```

(*End of definition for* `\l__talk_action_spec_str` *and* `\l__talk_frame_tagging_str`.)

1.4 Tagging for headers

__talk_header_tag_begin:n
__talk_header_tag_begin:e
__talk_header_tag_end:

Generalized control for inserting material into the header area (which is otherwise outside of tagging).

```
\cs_new_protected:Npn \__talk_header_tag_begin:n #1
  {
    \tag_resume:n { header }
    \tag_mc_end:
    \tag_struct_begin:n {#1}
    \tag_mc_begin:n { }
  }
\cs_generate_variant:Nn \__talk_header_tag_begin:n { e }
```

```
\cs_new_protected:Npn \__talk_header_tag_end:
  {
    \tag_mc_end:
    \tag_struct_end:
    \tag_mc_begin:n { artifact }
    \tag_suspend:n { header }
  }
```

(*End of definition for* `\__talk_header_tag_begin:n` *and* `\__talk_header_tag_end:`.)

1.5 Wallpaper

`\l__talk_footelem_left_skip`
`\l__talk_footelem_right_skip`
`\l__talk_footelem_color_tl`
`\l__talk_footelem_font_tl`

```
\NewTemplateType { footer-element } { 1 }
\DeclareTemplateInterface { footer-element } { talk } { 1 }
  {
    color        : tokenlist ,
    font         : tokenlist = ,
    left-hspace  : length = 0em ,
    right-hspace : length = 0em
  }
\DeclareTemplateCode { footer-element } { talk } { 1 }
  {
    color        = \l__talk_footelem_color_tl ,
    font         = \l__talk_footelem_font_tl ,
    left-hspace  = \l__talk_footelem_left_skip ,
    right-hspace = \l__talk_footelem_right_skip
  }
  {
    \tl_if_empty:nF {#1}
      {
        \hspace { \l__talk_footelem_left_skip }
        \group_begin:
          \tl_if_empty:NF \l__talk_footelem_color_tl
            { \color_select:V \l__talk_footelem_color_tl }
          \l__talk_footelem_font_tl
          #1
        \group_end:
        \hspace { \l__talk_footelem_right_skip }
      }
  }
\DeclareInstance { footer-element } { date } { talk } { }
\DeclareInstance { footer-element } { author } { talk } { }
\DeclareInstance { footer-element } { title } { talk } { }
\DeclareInstance { footer-element } { subtitle } { talk } { }
\DeclareInstance { footer-element } { institute } { talk } { }
\DeclareInstance { footer-element } { framenumber } { talk } { }
```

(*End of definition for* `\l__talk_footelem_left_skip` *and others.*)

`\l__talk_header_bg_tl`
`\l__talk_header_fg_tl`
`\l__talk_header_font_tl`
`\l__talk_header_ht_dim`
`\l__talk_header_left_skip`
`\l__talk_header_frametitle_bool`
`\l__talk_header_right_skip`

Templates for the header area. The background always covers the full width, but the text area may be narrower. The setup here aims to avoid repeated kerns but also dealing with complex conditionals, hence we always move to the edge of the paper first then adjust as required.

```
\NewTemplateType { header } { 0 }
\DeclareTemplateInterface { header } { talk } { 0 }
  {
    background-color  : tokenlist,
    color             : tokenlist = structure ,
    font              : tokenlist = \normalfont ,
    height            : length = \Gm@tmargin + \headsep ,
    left-hspace       : skip = \Gm@lmargin ,
    print-frame-title : boolean = true ,
    right-hspace      : skip = \Gm@rmargin
  }
\DeclareTemplateCode { header } { talk } { 0 }
  {
    background-color  = \l__talk_header_bg_tl ,
    color             = \l__talk_header_fg_tl ,
    font              = \l__talk_header_font_tl ,
    height            = \l__talk_header_ht_dim ,
    left-hspace       = \l__talk_header_left_skip ,
    print-frame-title = \l__talk_header_frametitle_bool ,
    right-hspace      = \l__talk_header_right_skip
  }
  {
    \noindent
    \__talk_wallpaper_hrule:Nnn
      \l__talk_header_bg_tl
      { \l__talk_header_ht_dim - \headsep }
      { \headsep }
    \skip_horizontal:n { \l__talk_header_left_skip }
    \group_begin:
      \tl_if_empty:NF \l__talk_header_fg_tl
        { \color_select:V \l__talk_header_fg_tl }
      \l__talk_header_font_tl
      \bool_if:NT \l__talk_header_frametitle_bool
        {
          \ExpandArgs { nnV }
            \UseInstance { frametitle } { header }
              \g__talk_frame_title_tl
        }
    \group_end:
  }
\DeclareInstance { header } { std } { talk } { }
\AddToHook { begindocument }
  {
    \DeclareInstanceCopy { header } { wallpaper } { std }
    \EditInstance { header } { wallpaper }
      { print-frame-title = false }
  }
```

(*End of definition for* `\l__talk_header_bg_tl` *and others.*)

`\l__talk_footer_bg_tl`
`\l__talk_footer_fg_tl`
`\l__talk_footer_font_tl`
`\l__talk_footer_order_clist`
`\l__talk_footer_sep_tl`
`\l__talk_footer_left_skip`
`\l__talk_footer_right_skip`

Templates for the footer area. Again the margins are handled in stages: here we do have a box for the content so the right skip is used, and we avoid an overfull box by including consideration of the right margin of the page layout.

```
\NewTemplateType { footer } { 0 }
```

```
\DeclareTemplateInterface { footer } { talk } { 0 }
  {
    background-color : tokenlist ,
    color            : tokenlist ,
    element-order    : commalist ,
    font             : tokenlist = \tiny ,
    left-hspace      : length = \Gm@lmargin ,
    right-hspace     : length = \Gm@rmargin ,
    separator        : tokenlist = \hfil
  }
\DeclareTemplateCode { footer } { talk } { 0 }
  {
    background-color = \l__talk_footer_bg_tl ,
    color            = \l__talk_footer_fg_tl ,
    element-order    = \l__talk_footer_order_clist ,
    separator        = \l__talk_footer_sep_tl ,
    font             = \l__talk_footer_font_tl ,
    left-hspace      = \l__talk_footer_left_skip ,
    right-hspace     = \l__talk_footer_right_skip
  }
  {
    \noindent
    \__talk_wallpaper_hrule:Nnn
      \l__talk_footer_bg_tl
      { \footskip }
      { \Gm@bmargin - \footskip }
    \skip_horizontal:n { \l__talk_footer_left_skip }
    \vbox_set_to_wd:Nnn \l__talk_tmp_box
      {
          \paperwidth
        - \l__talk_footer_left_skip
        - \l__talk_footer_right_skip
      }
      {
        \tl_if_empty:NF \l__talk_footer_fg_tl
          { \color_select:V \l__talk_footer_fg_tl }
        \l__talk_footer_font_tl
        \clist_pop:NNT \l__talk_footer_order_clist \l__talk_tmp_tl
          {
            \ExpandArgs { nVv } \UseInstance { footer-element } \l__talk_tmp_tl
              { @ \__talk_metadata_name:n { \l__talk_tmp_tl } }
            \clist_map_inline:Nn \l__talk_footer_order_clist
              {
                \tl_if_empty:cF { @ \__talk_metadata_name:n { ##1 } }
                  {
                    \l__talk_footer_sep_tl
                    \ExpandArgs { nnv }
                      \UseInstance { footer-element } {##1}
                        { @ \__talk_metadata_name:n { ##1 } }
                  }
              }
          }
        \hfil
      }
```

```
    \box_use_drop:N \l__talk_tmp_box
    \skip_horizontal:n { \l__talk_footer_right_skip - \Gm@rmargin }
  }
\DeclareInstance { footer } { std } { talk } { }
\AddToHook { begindocument }
  {
    \DeclareInstanceCopy { footer } { wallpaper } { std }
    \EditInstance { footer } { wallpaper }
      { element-order = }
  }
```

(*End of definition for* `\l__talk_footer_bg_tl` *and others.*)

`\__talk_metadata_name:n` A simple auxiliary to shorten metadata names but not the frame number. This is used as all the standard metadata is stored in two forms, and other than on the title page we use the shorter version. Full expansion is applied as this avoids any issue with stored names.

```
\cs_new:Npn \__talk_metadata_name:n #1
  {
    \str_if_eq:eeTF {#1} { framenumber }
      {#1}
      { short #1 }
  }
```

(*End of definition for* `\__talk_metadata_name:n`.)

`\__talk_wallpaper_hrule:Nnn` A simple abstraction for the top and bottom rules on the page.

```
\cs_new_protected:Npn \__talk_wallpaper_hrule:Nnn #1#2#3
  {
    \skip_horizontal:n { -\Gm@lmargin }
    \tl_if_empty:NF #1
      {
        \group_begin:
          \color_select:V #1
          \rule:nnn { \paperwidth } {#2} {#3}
          \skip_horizontal:n { -\paperwidth }
        \group_end:
      }
  }
```

(*End of definition for* `\__talk_wallpaper_hrule:Nnn`.)

`\ps@plain` `\ps@wallpaper` `\ps@talk` Install a standard header and footer template, and redefine the `plain` one as this will be used for frames without "wallpaper" which still need core links, *etc.* We also provide a version that only shows the visual elements: this is deliberately using the same settings as the main templates.

```
\cs_set_nopar:Npn \ps@plain
  {
    \cs_set_nopar:Npn \@oddhead
      {
        \hfil
      }
    \cs_set_nopar:Npn \@oddfoot { }
    \cs_set_eq:NN \@evenhead \@oddhead
```

```
    \cs_set_eq:NN \@evenfoot \@oddfoot
  }
\cs_set_nopar:Npn \ps@wallpaper
  {
    \cs_set_nopar:Npn \@oddhead
      {
        \UseInstance { header } { wallpaper }
        \hfil
      }
    \cs_set_nopar:Npn \@oddfoot
      {
        \UseInstance { footer } { wallpaper }
        \hfil
      }
    \cs_set_eq:NN \@evenhead \@oddhead
    \cs_set_eq:NN \@evenfoot \@oddfoot
  }
\cs_new_nopar:Npn \ps@talk
  {
    \cs_set_nopar:Npn \@oddhead
      {
        \UseInstance { header } { std }
        \hfil
      }
    \cs_set_nopar:Npn \@oddfoot { \UseInstance { footer } { std } }
    \cs_set_eq:NN \@evenhead \@oddhead
    \cs_set_eq:NN \@evenfoot \@oddfoot
  }
\pagestyle { talk }
```

(*End of definition for* `\ps@plain`, `\ps@wallpaper`, *and* `\ps@talk`. *These functions are documented on page* ??.)

1.6 The `frame` environment

`\l__talk_frame_bool` To track whether we are inside a frame or not.

```
\bool_new:N \l__talk_frame_bool
```

(*End of definition for* `\l__talk_frame_bool`.)

`\g__talk_frame_tag_bool` To track when a frame is being tagged: mainly needed for the header (and as a result global).

```
\bool_new:N \g__talk_frame_tag_bool
```

(*End of definition for* `\g__talk_frame_tag_bool`.)

`\l__talk_frame_verb_bool` Indicates that material was collected verbatim (and thus needs rescanning).

```
\bool_new:N \l__talk_frame_verb_bool
```

(*End of definition for* `\l__talk_frame_verb_bool`.)

`\g__talk_frame_int` The overall frame number, including LaTeX counter-like access.

`\c@frame` `\theframe` `\@framenumber`

```
\int_new:N \g__talk_frame_int
\cs_new_eq:NN \c@frame \g__talk_frame_int
\cs_new:Npn \theframe { \@arabic \c@frame }
\cs_new:Npn \@framenumber { \arabic { frame } }
```

(*End of definition for* `\g__talk_frame_int` *and others. These variables are documented on page* **??**.)

The total frames can be handled using the kernel properties.

```
\property_new:nnnn { totalframes } { shipout } { -1 }
  { \int_use:N \g__talk_frame_int }
\AddToHook { enddocument / afterlastpage }
  { \property_record:nn { lastpage } { totalframes } }
```

`\__talk_latexe_frame:n` As we will need to re-define `\frame` but have it available inside frames, a copy is made here.

```
\NewCommandCopy \__talk_latexe_frame:n \frame
```

(*End of definition for* `\__talk_latexe_frame:n`.)

`\__talk_frame_process:nn` Here, the frame content is received as the argument.

```
\cs_new_protected:Npn \__talk_frame_process:nn #1#2
  {
    \int_gincr:N \g__talk_frame_int
    \bool_set_true:N \l__talk_frame_bool
    \__talk_slide:nn {#1} {#2}
  }
```

(*End of definition for* `\__talk_frame_process:nn`.)

`\__talk_frame_tag:n` Wraps some content in tagging for a frame: we may have multiple of these in one logical frame, but that is non-standard.

```
\cs_new_protected:Npn \__talk_frame_tag:n #1
  {
    \tag_struct_begin:n { tag = frame }
    \int_gset:Nn \g__talk_frame_struct_int { \tag_get:n { struct_num } }
    \bool_gset_true:N \g__talk_frame_tag_bool
    #1
    \tag_struct_end:
  }
```

(*End of definition for* `\__talk_frame_tag:n`.)

`\__talk_frame_notag:n` The alternative: turn off tagging and suppress the function that would tag the frame title.

```
\cs_new_protected:Npn \__talk_frame_notag:n #1
  {
    \tag_mc_begin:n { artifact }
    \tag_suspend:n { frame }
    \bool_gset_false:N \g__talk_frame_tag_bool
    #1
    \par
    \tag_resume:n { frame }
    \tag_mc_end:
  }
```

(*End of definition for* `\__talk_frame_notag:n`.)

`frame` `frame*` The definition for the `frame` and `frame*` environments: the exact interface at both the document and code levels is still open.

```
\bool_if:NTF \l__talk_frame_title_bool
  {
    \RenewDocumentEnvironment { frame }
      { D <> { all } = { action-spec } O { } +m +b }
      {
        \keys_set:nn { talk / frame } {#2}
        \bool_set_false:N \l__talk_frame_verb_bool
        \__talk_frame_process:nn {#1} { \frametitle {#3} #4 }
      }
      { }
    \NewDocumentEnvironment { frame* }
      { D <> { all } = { action-spec } O { } +m c }
      {
        \keys_set:nn { talk / frame } {#2}
        \bool_set_true:N \l__talk_frame_verb_bool
        \tl_gset:Nn \g__talk_frame_title_tl {#3}
        \exp_args:Nne \__talk_frame_process:nn {#1}
          { \tl_to_str:n { \frametitle } \exp_not:n { {#3} #4 } }
      }
      { }
  }
  {
    \RenewDocumentEnvironment { frame }
      { !D <> { all } = { action-spec } !O { } +b }
      {
        \keys_set:nn { talk / frame } {#2}
        \bool_set_false:N \l__talk_frame_verb_bool
        \__talk_frame_process:nn {#1} {#3}
      }
      { }
    \NewDocumentEnvironment { frame* }
      { !D <> { all } = { action-spec } !O { } c }
      {
        \keys_set:nn { talk / frame } {#2}
        \bool_set_true:N \l__talk_frame_verb_bool
        \__talk_frame_process:nn {#1} {#3}
      }
      { }
  }
```

(*End of definition for* `frame` *and* `frame*`*. These functions are documented on page* **??**.)

⟨/class⟩

Part V
ltx-talk-frame – The structure of frames

1 ltx-talk-frame-structure implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

1.1 Columns

```
3 \keys_define:nn { talk }
4   { columns .inherit:n = talk / column }
```

`\l__talk_columns_wd_tl` We store the requested width for columns in a `tl` as this means that the key value will make sense even if it depends on the current `\textwidth`.

```
5 \keys_define:nn { talk / columns }
6   { width .tl_set:N = \l__talk_columns_wd_tl }
7 \keys_set:nn { talk / columns }
8   { width = \textwidth }
```

(*End of definition for* `\l__talk_columns_wd_tl`.)

`columns` (*env.*) Columns are block-like environments so we start and end with a `\par` to ensure correct tagging.

```
9  \NewDocumentEnvironment { columns } { D <> { all } O { } }
10   {
11     \__talk_action_begin:n {#1}
12     \par
13     \keys_set:nn { talk / columns } {#2}
14     \hbox_set_to_wd:Nnw \l__talk_tmp_box { \l__talk_columns_wd_tl }
15       \dim_set:Nn \textwidth { \l__talk_columns_wd_tl }
16       \dim_set_eq:NN \columnwidth \textwidth
17       \hfil
18       \ignorespaces
19   }
20   {
21       \unskip
22       \hfil
23     \hbox_set_end:
24     \box_use_drop:N \l__talk_tmp_box
25     \par
26     \__talk_action_end:
27   }
```

\l__talk_column_alignment_tl

```
\keys_define:nn { talk / column }
  {
    b .meta:n =
      { vertical-alignment = bottom } ,
    b .value_forbidden:n = true ,
    c .meta:n =
      { vertical-alignment = center } ,
    c .value_forbidden:n = true ,
    t .meta:n =
      { vertical-alignment = top } ,
    t .value_forbidden:n = true ,
    vertical-alignment .choices:nn =
      { bottom , center , top }
      {
        \tl_set_eq:NN \l__talk_column_alignment_tl
          \l_keys_value_tl
      }
  }
\keys_set:nn { talk / column }
  {
    vertical-alignment = center
  }
```

(*End of definition for* `\l__talk_column_alignment_tl`.)

__talk_column_align_bottom:n
__talk_column_align_center:n
__talk_column_align_top:n

Based on ideas in the highly experimental xbox.

```
\cs_new_protected:Npn \__talk_column_align_bottom:n #1
  { \vbox:n {#1} }
\cs_new_protected:Npn \__talk_column_align_center:n #1
  {
    \vbox:n
      {
        \hbox:n
          {
            \box_move_down:nn
              {
                  0.5 \box_ht:N \l__talk_tmp_box
                - \tex_fontdimen:D 22 ~ \tex_textfont:D 2 ~
              }
              { \vbox:n {#1} }
          }
      }
  }
\cs_new_protected:Npn \__talk_column_align_top:n #1
  { \vbox_top:n {#1} }
```

(*End of definition for* `\__talk_column_align_bottom:n`, `\__talk_column_align_center:n`, *and* `\__talk_column_align_top:n`.)

`column` (*env.*) A cut-down version of a minipage: we want to be clear on the semantic meaning. the action is applied inside the box after starting horizontal mode to avoid spacing issues when switching whatsits in and out.

```
\NewDocumentEnvironment { column } { D <> { all } O { } m }
```

```
  {
    \par
    \keys_set:nn { talk / column } {#2}
    \vbox_set_to_wd:Nnw \l__talk_tmp_box {#3}
      \dim_set:Nn \textwidth {#3}
      \dim_set_eq:NN \columnwidth \textwidth
      \@parboxrestore
      \leavevmode
      \raggedright
      \__talk_action_begin:n {#1}
      \ignorespaces
  }
```

The `\@ignore` here means that any spaces after `\end{column}` are suppressed by a `\ignorespaces` inserted by the kernel. The `\par` before `\__talk_action_end:` is needed as the group formed for actions would otherwise trap for example alignment changes.

```
  {
      \par
      \__talk_action_end:
    \vbox_set_end:
    \use:c { __talk_column_align_ \l__talk_column_alignment_tl :n }
      { \vbox_unpack_drop:N \l__talk_tmp_box }
    \hfil
    \par
    \@ignoretrue
  }
```

1.2 Floats

Well really "not floats at all" but the idea is clear.

`\l__talk_float_alignment_tl`

We only worry about horizontal alignment here.

```
\tl_new:N  \l__talk_float_alignment_tl
```

(*End of definition for* `\l__talk_float_alignment_tl`.)

A bit similar to the current approach to lists: we need a template at the start but a common function at the end. The `float-placement` key is at present just there to allow mopping up of any argument that is given by accident, hence maps to a temporary variable.

```
\NewTemplateType { floatenv } { 2 }
\DeclareTemplateInterface { floatenv } { talk } { 2 }
  {
    float-placement : tokenlist ,
    horizontal-alignment : choice { left , center ,  right } = left
  }
\DeclareTemplateCode { floatenv } { talk } { 2 }
  {
    float-placement = \l__talk_tmp_tl ,
    horizontal-alignment =
      {
        left = \tl_set:Nn \l__talk_float_alignment_tl { flushleft } ,
        center = \tl_set:Nn \l__talk_float_alignment_tl { center } ,
        right = \tl_set:Nn \l__talk_float_alignment_tl { flushright }
      }
```

```
  }
  {
    \SetTemplateKeys { floatenv } { talk } {#1}
    \begin { minipage } { \columnwidth }
      \begin { \l__talk_float_alignment_tl }
        \cs_set_nopar:Npn \@captype {#2}
  }
\DeclareInstance { floatenv } { std } { talk } { horizontal-alignment = left }
```

`\endfloatenv` And the common end function.

```
\cs_new_protected:Npn \endfloatenv
  {
      \end { \l__talk_float_alignment_tl }
    \end { minipage }
  }
```

(*End of definition for* `\endfloatenv`*. This function is documented on page* ??.)

`figure` (*env.*) `table` (*env.*) Unlike beamer, we allow for overlays for the environments as a whole.

```
\clist_map_inline:nn { figure , table }
  {
    \NewDocumentEnvironment {#1} { D <> { all } = { float-placement } O { } }
      {
        \__talk_action_begin:n {##1}
        \UseInstance { floatenv } { std } {##2} {#1}
      }
      {
        \endfloatenv
        \__talk_action_end:
      }
```

`\c@figure` `\thefigure` `\c@table` `\thetable` `\figurename` `\tableename` `\fnum@figure` `\fnum@table` The standard variables needed to make captions work (nothing for list of floats, as at present those are not offered).

```
    \newcounter {#1}
    \tl_new:c { #1 name }
    \tl_set:ce { #1 name } { \text_titlecase_first:n {#1} }
    \tl_new:c { fnum@ #1 }
    \tl_set:ce { fnum@ #1 }
      { \exp_not:c { #1 name } \exp_not:N \nobreakspace \exp_not:c { the #1 } }
  }
```

(*End of definition for* `\c@figure` *and others. These variables are documented on page* ??.)

The spacing values needed for the standard function.

```
\newlength \abovecaptionskip
\newlength \belowcaptionskip
\setlength \abovecaptionskip { 7pt }
\setlength \belowcaptionskip { 7pt }
```

`\@caption` This is a copy of the kernel version of the function, but with writing to the list of whatever file removed. It is very likely this needs to be reworked as a template, but that will likely come from the kernel.

```
\cs_set_protected:Npn \@caption #1 [ #2 ] #3
  {
    \par
```

```
    \begingroup
      \@parboxrestore
      \if@minipage \@setminipage \fi
      \normalsize
      \@makecaption { \csname fnum@ #1 \endcsname } { \ignorespaces #3 }
      \par
    \endgroup
  }
```

(*End of definition for* `\@caption`*. This function is documented on page* ??.)

1.3 Footnotes

`\@makefntext` A copy of the version provided by article: as for `\@caption`, we likely want a template here. It's not at present completely clear what will happen in the kernel (as the footnote templates currently leave `\@makefntext` alone).

```
\cs_new_protected:Npn \@makefntext #1
  {
    \parindent 1em
    \noindent
    \hb@xt@ 1.8em { \hss \@makefnmark }
    #1
  }
```

(*End of definition for* `\@makefntext`*. This function is documented on page* ??.)

⟨/class⟩

Part VI
ltx-talk-mode – Modes

1 ltx-talk-mode implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

__talk_mode:nT A simplified version of \mode: only deal with the argument form, only check the entire overlay spec as a string.

```
3 \prg_new_protected_conditional:Npnn \__talk_mode:n #1 { T }
4   {
5     \bool_lazy_or:nnTF
6       { \str_if_eq_p:nn {#1} { all } }
7       { \str_if_eq_p:Vn \l__talk_mode_str {#1} }
8       \prg_return_true:
9       \prg_return_false:
10  }
```

(*End of definition for* __talk_mode:nT.)

\mode

```
11 \NewDocumentCommand \mode { D <> { all } +m }
12   { \__talk_mode:nT {#1} {#2} }
```

(*End of definition for* \mode. *This function is documented on page* ??.)

```
13 ⟨/class⟩
```

Part VII
ltx-talk-overlay – Overlays

1 ltx-talk-overlay implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

1.1 Utilities

__talk_if_overlay:nTF
__talk_if_overlay:VTF
__talk_overlay_arg:n

```
3 \prg_new_protected_conditional:Npnn \__talk_if_overlay:n #1 { T , F , TF }
4   {
5     \__talk_decode_parse:n {#1}
6     \bool_if:NTF \l__talk_decode_overlays_bool
7       \prg_return_true:
8       \prg_return_false:
9   }
10 \prg_generate_conditional_variant:Nnn \__talk_if_overlay:n { V } { T , F , TF }
```

A macro processor variant of the check that always results in an N-type bool.

```
11 \cs_new_protected:Npn \__talk_overlay_arg:n #1
12   {
13     \__talk_if_overlay:nTF {#1}
14       { \cs_set:Npn \ProcessedArgument { \c_true_bool } }
15       { \cs_set:Npn \ProcessedArgument { \c_false_bool } }
16   }
```

(*End of definition for* `\__talk_if_overlay:nTF` *and* `\__talk_overlay_arg:n`.)

1.2 Action commands and environments

Commands that can be used as actions all have a common form (with one exception). The common internal structure is used to enable them to be used as actions by looking for the name `\__talk_action_`⟨*name*⟩`:N`. This is set up such that the inactive versions insert a whatsit equal to that which would be present if they were active: that's needed for spacing.

__talk_action_:N

The fallback action. At present, we need to create a whatsit here to avoid spacing issues. (In LuaTEX, if we can move to attributes, this can be removed.)

```
17 \cs_new_protected:Npn \__talk_action_:N #1 { \opacity_select:n { 1 } }
```

(*End of definition for* `\__talk_action_:N`.)

__talk_action_alert:N

At present a color selection.

```
18 \cs_new_protected:Npn \__talk_action_alert:N #1
19   {
20     \bool_if:NTF #1
21       { \color_select:n { alert } }
```

```
      { \color_select:n { . } }
  }
```

(*End of definition for* `\__talk_action_alert:N`.)

`\__talk_action_invisible:N` `\__talk_action_visible:N`

Simply hide unconditionally.

```
\cs_new_protected:Npn \__talk_action_invisible:N #1
  {
    \bool_if:NTF #1
      { \opacity_select:n { 0 } }
      { \opacity_select:n { 1 } }
  }
\cs_new_protected:Npn \__talk_action_visible:N #1
  {
    \bool_if:NTF #1
      { \opacity_select:n { 1 } }
      { \opacity_select:n { 0 } }
  }
```

(*End of definition for* `\__talk_action_invisible:N` *and* `\__talk_action_visible:N`.)

`\__talk_action_only_begin:N` `\__talk_action_only_end:N`

Here, we simply throw away the content we do not want: this is done by typesetting in a disposable box.

```
\cs_new_protected:Npn \__talk_action_only:N #1
  {
    \bool_if:NF #1
      { \vbox_set:Nw \l__talk_tmp_box }
  }
\cs_new_protected:Npn \__talk_action_only_end:N #1
  {
    \bool_if:NF #1
      { \vbox_set_end: }
  }
```

(*End of definition for* `\__talk_action_only_begin:N` *and* `\__talk_action_only_end:N`.)

`\l__talk_uncover_hidden_fp`

Currently just an on-off, but that will change.

```
\NewTemplateType { hidden } { 0 }
\DeclareTemplateInterface { hidden } { talk } { 0 }
  { opacity : real = 0 }
\DeclareTemplateCode { hidden } { talk } { 0 }
  { opacity = \l__talk_uncover_hidden_fp }
  { \opacity_select:n { \l__talk_uncover_hidden_fp } }
\DeclareInstance { hidden } { std } { talk } { }
```

(*End of definition for* `\l__talk_uncover_hidden_fp`.)

`\__talk_action_uncover:N`

Use the template

```
\cs_new_protected:Npn \__talk_action_uncover:N #1
  {
    \bool_if:NTF #1
      { \opacity_select:n { 1 } }
      { \UseInstance { hidden } { std } }
  }
```

(*End of definition for* `\__talk_action_uncover:N`.)

`\only` `\invisible` `\uncover` Commands and environments where the payload applies when the material is not active on the slide.

```
\clist_map_inline:nn { only , invisible , uncover }
  {
    \ExpandArgs { cne } \NewDocumentCommand {#1}
      { > { \__talk_overlay_arg:n } D <> { all } +m }
      {
        \group_begin:
          \exp_not:c { __talk_action_ #1 :N } ##1
          ##2
          \cs_if_exist:cT { __talk_action_ #1 _end:N }
            { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
        \group_end:
      }
```

(*End of definition for* `\only`, `\invisible`, *and* `\uncover`. *These functions are documented on page* **??**.)

`onlyenv` (*env.*) `invisibleenv` (*env.*) `uncoverenv` (*env.*)

```
    \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
      { > { \__talk_overlay_arg:n } D <> { all } }
      { \exp_not:c { __talk_action_ #1 :N } ##1 }
      {
        \cs_if_exist:cT { __talk_action_ #1 _end:N }
          { \exp_not:c { __talk_action_ #1 _end:N } ##1 }
      }
  }
```

`\alert` `\visible` And those where the action applies when we are on the slide.

```
\clist_map_inline:nn { alert , visible }
  {
    \ExpandArgs { cne } \NewDocumentCommand {#1}
      { > { \__talk_overlay_arg:n } D <> { all } +m }
      {
        \group_begin:
          \exp_not:c { __talk_action_ #1 :N } ##1
            ##2
         \group_end:
      }
```

(*End of definition for* `\alert` *and* `\visible`. *These functions are documented on page* **??**.)

`alertenv` (*env.*) `visibleenv` (*env.*)

```
    \ExpandArgs { nnee } \NewDocumentEnvironment { #1 env }
      { > { \__talk_overlay_arg:n } D <> { all } }
      { \exp_not:c { __talk_action_ #1 :N } ##1 }
      { }
  }
```

`\only` This code needs to be done manually as for the command version the content must be entirely discarded. That can't work for the environment version, which has to deal with for example single items in a list (and so cannot be collected up verbatim and must use a box).

```
\RenewDocumentCommand \only { D <> { all } +m }
  {
    \__talk_if_overlay:nT {#1}
      {#2}
  }
```

(*End of definition for* `\only`. *This function is documented on page* **??**.)

`\l__talk_saved_overlays_bool`
`\l__talk_saved_action_str`
`\l__talk_saved_actions_bool`

```
\bool_new:N \l__talk_saved_overlays_bool
\str_new:N \l__talk_saved_action_str
\bool_new:N \l__talk_saved_actions_bool
```

(*End of definition for* `\l__talk_saved_overlays_bool`, `\l__talk_saved_action_str`, *and* `\l__talk_saved_actions_bool`.)

`actionenv` (*environment*)
`\action` (*command*)
`\__talk_action_begin:n`
`\__talk_action_end:`

As we need data on not just overlays but also actions at the end of the environment, this has to be done manually. To allow working with environments but also items, the code needs to save data for the end function. The group is needed for cases where we are not in a LaTeX environment group.

```
\NewDocumentCommand \action { D <> { all } +m }
  {
    \group_begin:
      \__talk_action_begin:n {#1}
      #2
      \__talk_action_end:
    \group_end:
  }
\NewDocumentEnvironment { actionenv } { D <> { all } }
  { \__talk_action_begin:n {#1} }
  { \__talk_action_end: }
\cs_new_protected:Npn \__talk_action_begin:n #1
  {
    \group_begin:
      \__talk_decode_parse:n {#1}
      \bool_set_eq:NN \l__talk_saved_overlays_bool
        \l__talk_decode_overlays_bool
      \str_set_eq:NN \l__talk_saved_action_str
        \l__talk_decode_action_str
      \bool_set_eq:NN \l__talk_saved_actions_bool
        \l__talk_decode_actions_bool
      \bool_if:NTF \l__talk_decode_overlays_bool
        {
          \use:c { __talk_action_ \l__talk_decode_action_str :N }
            \l__talk_decode_actions_bool
        }
        { \UseInstance { hidden } { std } }
  }
\cs_new_protected:Npn \__talk_action_end:
  {
      \bool_if:NT \l__talk_saved_overlays_bool
        {
          \cs_if_exist_use:cF
            { __talk_action_ \l__talk_saved_action_str _end:N }
```

```
            { \use_none:n }
              \l__talk_saved_actions_bool
        }
    \group_end:
  }
```

(*End of definition for* `\action`, `\__talk_action_begin:n`, *and* `\__talk_action_end:`. *This function is documented on page* ??.)

1.3 Non-action commands and environments

This section contains commands and environments that do *not* need to be made available as actions.

`\alt` Simple wrappers around the internal switch.

```
\NewDocumentCommand \alt { D <> { all } +m +m }
  {
    \__talk_if_overlay:nTF {#1}
      {#2}
      {#3}
  }
```

(*End of definition for* `\alt`. *This function is documented on page* ??.)

`\onslide` `\__talk_onslide:n` `\__talk_onslide_reset:` Simply make transparent: we will likely need to save the original opacity level. To allow us to apply independent of group level, a little work is needed.

```
\NewDocumentCommand \onslide { D <> { all } }
  { \__talk_onslide:n {#1} }
\cs_new_protected:Npn \__talk_onslide:n #1
  {
    \tl_use:N \g__talk_onslide_tl
    \__talk_if_overlay:nTF {#1}
      { \__talk_onslide_reset: }
      {
        \opacity_select:n { 0 }
        \tl_gset:Nn \g__talk_onslide_escape_tl
          {
            \opacity_select:n { 0 }
            \group_insert_after:N \g__talk_onslide_escape_tl
          }
        \group_insert_after:N \g__talk_onslide_escape_tl
        \tl_gset:Nn \g__talk_onslide_tl
          {
            \tl_gclear:N \g__talk_onslide_tl
            \tl_gclear:N \g__talk_onslide_escape_tl
            \__talk_onslide_reset:
          }
      }
  }
\cs_new_protected:Npn \__talk_onslide_reset: { \opacity_select:n { 1 } }
```

(*End of definition for* `\onslide`, `\__talk_onslide:n`, *and* `\__talk_onslide_reset:`. *This function is documented on page* ??.)

\g__talk_onslide_tl
\g__talk_onslide_escape_tl

```
\tl_new:N \g__talk_onslide_tl
\tl_new:N \g__talk_onslide_escape_tl
```

(*End of definition for* `\g__talk_onslide_tl` *and* `\g__talk_onslide_escape_tl`.)

`\temporal` A tricky one: to separate the not-on-current-slide cases, the flag to continue is used.

```
\NewDocumentCommand \temporal { D <> { all } +m +m +m }
  {
    \__talk_if_overlay:nTF {#1}
      {#3}
      {
        \bool_if:NTF \g__talk_slide_continue_bool
          {#4}
          {#2}
      }
  }
```

(*End of definition for* `\temporal`. *This function is documented on page* ??.)

`\pause` A thin wrapper.

```
\NewDocumentCommand \pause { o }
  {
    \IfNoValueTF {#1}
      { \int_gincr:N \g__talk_pauses_int }
      { \int_gset:Nn \g__talk_pauses_int {#1} }
    \exp_args:Ne \__talk_onslide:n { \int_eval:n { \g__talk_pauses_int + 1 } - }
  }
```

(*End of definition for* `\pause`. *This function is documented on page* ??.)

1.4 Fixed-size areas

`\__talk_overprint_begin:n` A common auxiliary for overprinting, which starts off much the same for both `overlayarea` and `overprint`.

```
\cs_new_protected:Npn \__talk_overprint_begin:n #1
  {
    \par
    \vbox_set_to_wd:Nnw \l__talk_tmp_box {#1}
      \raggedright
      \ignorespaces
  }
```

(*End of definition for* `\__talk_overprint_begin:n`.)

`overlayarea` (*env.*) An initial approach: quite similar to a column.

```
\NewDocumentEnvironment { overlayarea } { m m }
  { \__talk_overprint_begin:n {#1} }
  {
    \vbox_set_end:
    \vbox_to_ht:nn {#2}
      {
        \box_use_drop:N \l__talk_tmp_box
        \vfil
```

```
      }
    \par
  }
```

`\l__talk_overprint_int` Track the overprints on a slide: as the slide forms a group, we do not need to worry about resetting.

```
\int_new:N \l__talk_overprint_int
```

(*End of definition for* `\l__talk_overprint_int`.)

`\__talk_frame_overprint:` To refer to the current overprint environment within the document: needed in the `.aux` so avoids using non-letters.

```
\cs_new:Npn \__talk_frame_overprint:
  {
    \int_to_Roman:n \g__talk_frame_int
    \int_to_roman:n \l__talk_overprint_int
  }
```

(*End of definition for* `\__talk_frame_overprint:`.)

`\__talk_overprint_begin:n` (*env.*) `\__talk_overprint_check_ht:n` For overprinting, in contrast to beamer we use a two-pass approach to save the size at the end of the run: this means you can use `\only` for example in overprinting.

```
\NewDocumentEnvironment { overprint } { O { \textwidth } }
  { \__talk_overprint_begin:n {#1} }
  {
    \vbox_set_end:
    \int_incr:N \l__talk_overprint_int
    \__talk_overprint_save_ht:
    \cs_if_exist:cTF
      { overprint@ \__talk_frame_overprint: }
      {
        \dim_compare:vNnTF
          { overprint@ \__talk_frame_overprint: }
            > { \box_ht:N \l__talk_tmp_box }
          {
            \vbox_to_ht:vn
              { overprint@ \__talk_frame_overprint: }
              {
                \box_use_drop:N \l__talk_tmp_box
                \vfil
              }
          }
          { \box_use_drop:N \l__talk_tmp_box }
      }
      { \box_use_drop:N \l__talk_tmp_box }
    \par
  }
```

As there is no clear end-point for overprinting, we need to be careful to keep the current width separate from the saved one. The rest is then about saving to the `.aux` file and helping out the user.

```
\cs_new_protected:Npn \__talk_overprint_save_ht:
  {
    \tl_if_exist:cF { g__talk_overprint_ \__talk_frame_overprint: _tl }
```

```
      {
        \tl_new:c { g__talk_overprint_ \__talk_frame_overprint: _tl }
        \tl_gset:cn { g__talk_overprint_ \__talk_frame_overprint: _tl }
          { 0pt }
      }
    \tl_gset:ce { g__talk_overprint_ \__talk_frame_overprint: _tl }
      {
        \dim_max:vn { g__talk_overprint_ \__talk_frame_overprint: _tl }
          { \box_ht:N \l__talk_tmp_box }
      }
    \legacy_if:nT { @filesw }
      {
        \iow_now:Ne \@auxout
          {
            \gdef \exp_not:c { overprint@ \__talk_frame_overprint: }
              {
                \exp_not:v { g__talk_overprint_ \__talk_frame_overprint: _tl }
              }
          }
      }
    \hook_gput_code:nne { enddocument / afterlastpage } { talk }
       { \__talk_overprint_check_ht:n { \__talk_frame_overprint: } }
  }
\cs_new_protected:Npn \__talk_overprint_check_ht:n #1
  {
    \bool_lazy_and:nnF
      { \exp_not:N \cs_if_exist_p:c { overprint@ #1 } }
      {
        \dim_compare_p:vNv { overprint@ #1 } = { g__talk_overprint_ #1 _tl }
      }
      {
        \msg_warning:nn { talk } { overprint-ht }
        \cs_gset_protected:Npn \__talk_overprint_check_ht:n ##1 { }
      }
  }
\msg_new:nnn { talk } { overprint-ht }
  {
    Overprint~area~height~has~changed:\\
    rerun~LaTeX.
  }
```

(*End of definition for* `\__talk_overprint_save_ht:` *and* `\__talk_overprint_check_ht:n`.)

1.5 Adding overlays to existing commands

`\textbf` `\textit` `\textmd` `\textnormal` `\textrm` `\textsc` `\textsf` `\textsl` `\texttt` `\textup` `\emph` `\stdtextbf` `\stdtextit` `\stdtextmd` `\stdtextnormal` `\stdtextrm` `\stdtextsc` `\stdtextsf` `\stdtextsl`

Make the standard text commands overlay-aware. To keep the spacing unchanged when the command is not active, we use the same approach as the kernel does for inserting the right grouping.

```
\tl_map_inline:nn
  {
    \textbf
    \textit
    \textmd
    \textnormal
```

```
    \textrm
    \textsc
    \textsf
    \textsl
    \texttt
    \textup
    \emph
  }
  {
    \ExpandArgs { c } \NewCommandCopy { std \cs_to_str:N #1 } #1
    \ExpandArgs { Nne } \RenewDocumentCommand #1
      { D <> { all } +m }
      {
        \exp_not:N \__talk_if_overlay:nTF {##1}
          { \exp_not:c { std \cs_to_str:N #1 } }
          { \exp_not:N \__talk_textcmd_eqiv:n }
            {##2}
      }
  }
\cs_new_protected:Npn \__talk_textcmd_eqiv:n #1
  {
    \mode_if_math:TF
      { { \mbox {#1} } }
      {
        \mode_leave_vertical:
        {#1}
      }
  }
```

(*End of definition for* `\textbf` *and others. These functions are documented on page* **??**.)

`\includegraphics` `\stdincludegraphics`

Just wrap up the args and forward if appropriate. The star is `#1` here as that matches the documented behavior of starred commands generally.

```
\RequirePackage { graphicx }
\NewCommandCopy \stdincludegraphics \includegraphics
\RenewDocumentCommand \includegraphics { s D <> { all } o o m }
  {
    \__talk_if_overlay:nT {#2}
      {
        \use:e
          {
            \exp_not:N \stdincludegraphics
              \IfBooleanT #1 { * }
              \IfNoValueF {#3} { [ \exp_not:n { {#3} } ] }
              \IfNoValueF {#4} { [ \exp_not:n { {#4} } ] }
          }
            {#5}
      }
  }
```

(*End of definition for* `\includegraphics` *and* `\stdincludegraphics`*. These functions are documented on page* **??**.)

`\label` `\__talk_label:n`

Here, we can't wrap the existing command up as we need the space hack, so it has to be declared from scratch. There is also a non-standard overlay default. At present, no

special tricks as seen in beamer.

```
\RenewDocumentCommand \label { D <> { 1 } m }
  {
    \@bsphack
    \__talk_if_overlay:nT {#1}
      { \__talk_label:n {#2} }
    \@esphack
  }
\cs_new_protected:Npn \__talk_label:n #1
  {
    \begingroup
      \UseHookWithArguments { label } { 1 } {#1}
      \protected@write \@auxout { }
        {
          \string \newlabel {#1}
            {
              { \@currentlabel }
              { \thepage }
              { \@currentlabelname }
              { \@currentHref }
              { \@kernel@reserved@label@data }
            }
        }
    \endgroup
  }
```

(*End of definition for* `\label` *and* `\__talk_label:n`*. This function is documented on page* **??**.)

⟨/class⟩

Part VIII
ltx-talk-required – "Required" definitions

1 ltx-talk-required implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

Here we collect up things that are more-or-less required to create a useful class but are not defined by the LaTeX kernel for historical reasons. They are therefore largely copies from `article.cls` and contain "classical" definitions so that they follow the expectations of third-party code.

`\today` This is the definition as done in the standard classes.

```
3 \cs_new_nopar:Npn \today
4   {
5     \ifcase \month \or
6       January \or
7       February \or
8       March \or
9       April \or
10      May \or
11      June \or
12      July \or
13      August \or
14      September \or
15      October \or
16      November \or
17      December
18    \fi
19    \space
20    \number \day ,
21    \space
22    \number \year
23  }
```

(*End of definition for* `\today`. *This function is documented on page* ??.)

1.1 Standard design settings

```
24 \setcounter { tocdepth } { 3 }
25 \setlength \arraycolsep { 5pt }
26 \setlength \tabcolsep { 6pt }
27 \setlength \arrayrulewidth { 0.4pt }
28 \setlength \doublerulesep { 2pt }
29 \setlength \tabbingsep { \labelsep }
30 \skip \@mpfootins = \skip \footins
```

```
\setlength \fboxsep { 3pt }
\setlength \fboxrule { 0.4pt }
```

1.2 List support

```
\setlength  \labelsep  { 0.5em }
\cs_new:Npn \labelenumi { \theenumi . }
\cs_new:Npn \labelenumii { ( \theenumii ) }
\cs_new:Npn \labelenumiii { \theenumiii . }
\cs_new:Npn \labelenumiv { \theenumiv . }
\cs_new:Npn \labelitemi  { \labelitemfont \textbullet }
\cs_new:Npn \labelitemii { \labelitemfont \bfseries \textendash }
\cs_new:Npn \labelitemiii { \labelitemfont \textasteriskcentered }
\cs_new:Npn \labelitemiv  { \labelitemfont \textperiodcentered }
\cs_new:Npn \labelitemfont { \normalfont }
```

```
\setlength \leftmargini   { 2em }
\setlength \leftmarginii  { 2em }
\setlength \leftmarginiii { 2em }
\setlength  \labelsep  { 0.5em }
\setlength  \labelwidth { \leftmargini }
\addtolength \labelwidth { -\labelsep }
\cs_gset_nopar:Npn \@listi
  {
    \leftmargin \leftmargini
    \topsep 3pt plus 2pt minus 2.5pt
    \parsep 0pt
    \itemsep 3pt plus 2pt minus 3pt
  }
\cs_gset_eq:NN \@listI \@listi
\cs_gset_nopar:Npn \@listii
  {
    \leftmargin \leftmarginii
    \topsep    2pt plus 1pt minus 2pt
    \parsep    0pt plus 1pt
    \itemsep   \parsep
  }
\cs_gset_nopar:Npn \@listiii
  {
    \leftmargin \leftmarginiii
    \topsep    2pt plus 1pt minus 2pt
    \parsep    0pt plus 1pt
    \itemsep   \parsep
  }
\setlength \partopsep { 0pt }
```

⟨/class⟩

Part IX
ltx-talk-structure – Structural commands

1 ltx-talk-structure implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

1.1 Frame title

\g__talk_frame_title_tl
\g__talk_frame_subtitle_tl

```
3 \tl_new:N \g__talk_frame_title_tl
4 \tl_new:N \g__talk_frame_subtitle_tl
```

(*End of definition for* \g__talk_frame_title_tl *and* \g__talk_frame_subtitle_tl.)

\frametitle

Just data storage: at the present no use of the optional argument.

```
5 \NewDocumentCommand \frametitle { D <> { all } O {#3} m }
6   {
7     \__talk_if_overlay:nT {#1}
8       { \tl_gset:Nn \g__talk_frame_title_tl {#3} }
9   }
10 \NewDocumentCommand \framesubtitle { D <> { all } O {#3} m }
11   {
12     \__talk_if_overlay:nT {#1}
13       { \tl_gset:Nn \g__talk_frame_subtitle_tl {#3} }
14   }
```

(*End of definition for* \frametitle. *This function is documented on page* ??.)

__talk_frame_title:n
__talk_frame_title_tagged:n

Inserting the frame title requires we deal with tagging as well as appearance: if there is a title, we need to tag just this part of the header.

```
15 \NewTemplateType { frametitle } { 1 }
16 \DeclareTemplateInterface { frametitle } { talk } { 1 }
17   {
18     after-vspace  : skip = \bigskipamount ,
19     before-vspace : skip = 0em ,
20     color         : tokenlist = ,
21     font          : tokenlist = \Large \bfseries
22   }
23 \DeclareTemplateCode { frametitle } { talk } { 1 }
24   {
25     after-vspace  = \l__talk_frametitle_after_skip ,
26     before-vspace = \l__talk_frametitle_before_skip ,
27     color         = \l__talk_frametitle_color_tl ,
28     font          = \l__talk_frametitle_font_tl
29   }
```

```
  {
    \noindent
    \vspace { \l__talk_frametitle_before_skip }
    \group_begin:
      \tl_if_empty:NF \l__talk_frametitle_color_tl
        { \color_select:V \l__talk_frametitle_color_tl }
      \l__talk_frametitle_font_tl
      \tl_if_blank:nF {#1}
        { \__talk_frame_title:n {#1} }
      \par
    \group_end:
    \vspace { \l__talk_frametitle_after_skip }
  }
\DeclareInstance { frametitle } { header } { talk } { }
\cs_new_protected:Npn \__talk_frame_title:n #1
  {
    \bool_if:NTF \g__talk_frame_tag_bool
      { \__talk_frame_title_tagged:n }
      { \use:n }
        {#1}
  }
\cs_new_protected:Npn \__talk_frame_title_tagged:n #1
  {
   \__talk_header_tag_begin:e
      {
        firstkid = true ,
        parent   = \int_use:N \g__talk_frame_struct_int ,
        tag      = frametitle ,
        title    = { \text_purify:n { \g__talk_frame_title_tl } } ,
      }
    \group_begin:
      \tagpdfparaOff
      #1
    \group_end:
    \__talk_header_tag_end:
  }
```

(*End of definition for* `\__talk_frame_title:n` *and* `\__talk_frame_title_tagged:n`.)

1.2 Sectioning

`\l__talk_section_tl`
`\g__talk_section_tl`
`\l__talk_subsection_tl`
`\g__talk_subsection_tl`
`\l__talk_subsubsection_tl`
`\g__talk_subsubsection_tl`

Two versions of the data store: one set locally (but at the top level) for general use, one set (and more importantly cleared) globally to allow insertion in the header area just once per name.

```
\tl_new:N \l__talk_section_tl
\tl_new:N \g__talk_section_tl
\tl_new:N \l__talk_subsection_tl
\tl_new:N \g__talk_subsection_tl
\tl_new:N \l__talk_subsubsection_tl
\tl_new:N \g__talk_subsubsection_tl
```

(*End of definition for* `\l__talk_section_tl` *and others.*)

`\section`
`\subsection`
`\subsubsection`
`\thesection`
`\thesubsection`
`\thesubsubsection`

Here, we need full LaTeX counters, so create them using the appropriate mechanism: that also means we can sort out counter dependency and the appearance (using the same setup

as in article). As (subsub)section numbers never increment inside frames, we remove these counters from the general tracker.

```
\newcounter { section }
\newcounter { subsection } [ section ]
\newcounter { subsubsection } [ subsection ]
\seq_gremove_all:Nn \l__talk_cnt_reset_seq { section }
\seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsection }
\seq_gremove_all:Nn \l__talk_cnt_reset_seq { subsubsection }
\cs_gset:Npn \thesection { \@arabic \c@section }
\cs_gset:Npn \thesubsection { \thesection . \@arabic \c@subsection }
\cs_gset:Npn \thesubsubsection { \thesubsection . \@arabic \c@subsubsection }
```

(*End of definition for* `\section` *and others. These functions are documented on page* **??**.)

`\section`
`\subsection`
`\subsubsection`
`\insertsection`
`\insertsubsection`
`\insertsubsubsection`

The sectioning commands all have essentially the same form: we therefore create using a generator with the necessary conditionals in place. As we do not typeset sections at this stage, the code is quite different from article. This also means that the bookmark links need to point forward to the next slide: if that doesn't appear, the bookmarks will be out. Using the general scratch sequence here should be OK: it really is a one-off setting. We need a sequence to allow indexed mapping to avoid any extra setup for the depth value.

```
\seq_set_from_clist:Nn \l_tmpa_seq
  { section , subsection , subsubsection }
\seq_map_indexed_inline:Nn \l_tmpa_seq
  {
    \use:e
      {
        \NewDocumentCommand \exp_not:c { insert #2 } { }
          {
            \exp_not:N \tl_use:N
            \exp_not:c { l__talk_ #2 _tl }
          }
        \NewDocumentCommand \exp_not:c {#2}
          { s D <> { all } O {##4} m }
          {
            \exp_not:N \refstepcounter {#2}
            \UseTaggingSocket { sec / end } { \use:c { toclevel@ #2 } }
            \UseTaggingSocket { sec / begin }
              {
                { \use:c { toclevel@ #2 } }
                {
                  tag =
                    \exp_not:N \UseStructureName
                      { sec / \use:c { toclevel@ #2 } }
                }
              }
            \tl_set:Nn \exp_not:c { l__talk_ #2 _tl } {##4}
            \UseTaggingSocket { talk / sec / title } {#2}
            \str_if_eq:nnT {#2} { section }
              { \tl_clear:N \exp_not:N \l__talk_subsection_tl }
            \str_if_eq:nnF {#2} { subsubsection }
              { \tl_clear:N \exp_not:N \l__talk_subsubsection_tl }
            \exp_not:N \addcontentsline { toc } {#2}
```

```
            {
              \exp_not:N \int_compare:nNnF {#1} >
                { \exp_not:N \value { secnumdepth } }
                {
                  \exp_not:N \protect \exp_not:N \numberline
                    { \exp_not:c { the #2 } }
                }
              ##4
            }
          \hook_use:n { #2 / begin }
        }
      \hook_new:n { #2 / begin }
    }
}
```

(*End of definition for* `\section` *and others. These functions are documented on page* **??**.)

`talk/sec/title` The argument is one of section, subsection or subsubsection.

`\__talk_sect_tag:nn`

```
\NewTaggingSocket { talk / sec / title } { 1 }
\NewTaggingSocketPlug { talk / sec / title } { default }
  { \exp_args:Ne \__talk_sect_tag:nn { \text_purify:v { l__talk_ #1 _ tl } } {#1} }
\cs_new_protected:Npn \__talk_sect_tag:nn #1#2
  {
    \tag_struct_begin:e
      {
        tag        =
          \UseStructureName { sec / \use:c { toclevel@ #2 } / title } ,
        title      = {#1} ,
        actualtext = {#1} ,
      }
    \tag_struct_end:
  }
\AssignTaggingSocketPlug { talk / sec / title } { default }
```

(*End of definition for* `talk/sec/title` *and* `\__talk_sect_tag:nn`*. This function is documented on page* **??**.)

1.3 Table of contents

`\@starttoc` The standard kernel implementation here deliberately overwrites the file as soon as it's read. That's no good for us as the table of contents can be read multiple times. So we modify the code: we start from the tagging-aware version (this may need to be revisited). We retain the LaTeX 2_ε code as much as possible.

```
\cs_gset_protected:Npn \@starttoc #1
  {
    \begingroup
      \makeatletter
      \UseTaggingSocket { toc / starttoc / before } {#1}
      \@input { \jobname .#1 }
      \UseTaggingSocket { toc / starttoc / after } {#1}
      \legacy_if:nT { @filesw }
        {
          \AddToHook { enddocument / afterlastpage }
            {
```

```
              \expandafter \newwrite \csname tf@ #1 \endcsname
              \immediate \openout \csname tf@ #1 \endcsname \jobname .#1 \relax
            }
        }
      \@nobreakfalse
    \endgroup
  }
```

(*End of definition for* `\@starttoc`. *This function is documented on page* ??.)

`\tableofcontents` For the present simply print the output.

```
\NewDocumentCommand \tableofcontents { O { } }
  {
    \group_begin:
      \@starttoc { toc }
    \group_end:
  }
```

(*End of definition for* `\tableofcontents`. *This function is documented on page* ??.)

`\l@section` `\l@subsection` `\l@subsubsection` `\__talk_toc_aux:nnnn` `\__talk_toc_dest:n` `\__talk_toc_dest:w` `\__talk_toc_level:nnnn`

Initial hard-coded versions to be templated once we have some other effects also working. We may need to look at this "higher up" as we will need to know the section numbers.

```
\cs_new_protected:Npn \l@section #1#2
  { \__talk_toc_aux:nnnn { 1 } { \bfseries \color { structure } } {#1} {#2} }
\cs_new_protected:Npn \l@subsection #1#2
  {
    \__talk_toc_aux:nnnn
      { 2 }
      {
        \skip_set:Nn \leftskip { 2em }
        \color { . }
      }
      {#1} {#2}
  }
\cs_new_protected:Npn \l@subsubsection #1#2
  {
    \__talk_toc_aux:nnnn
      { 3 }
      {
        \skip_set:Nn \leftskip { 4em }
        \color { . }
        \footnotesize
      }
      {#1} {#2}
  }
\cs_new_protected:Npn \__talk_toc_aux:nnnn #1#2#3#4
  {
    \int_compare:nNnTF { \value { section } } < 1
      { \use:n }
      { \__talk_toc_dest:n }
        { \__talk_toc_level:nnnn {#1} {#2} {#3} {#4} }
  }
```

We can extract the details for the TOC levels from `\@contentsline@destination`. At present, that is quite simple-minded: if we are in the current section, show fully, else make semi-opaque. Needs a rounded-out interface but the basic idea will be the same.

```
\cs_new_protected:Npn \__talk_toc_dest:n
  {
    \exp_after:wN \__talk_toc_dest:w \@contentsline@destination
      . 0 . 0 . 0 . \q_stop
  }
\cs_new_protected:Npn \__talk_toc_dest:w #1 . #2 . #3 . #4 . #5 \q_stop #6
  {
    \int_compare:nNnTF { \value { section } } = {#2}
      {#6}
      {
        \group_begin:
          \opacity_select:n { 0.2 }
          #6
        \group_end:
      }
  }
\cs_new_protected:Npn \__talk_toc_level:nnnn #1#2#3#4
  {
    \int_compare:nNnF {#1} > { \value { tocdepth } }
      {
        \group_begin:
          \noindent
          #2
          \UseHookWithArguments { contentsline / text / before } { 4 }
            {#1} {#3} {#4} { \@contentsline@destination }
          #3
          \UseHookWithArguments { contentsline / text / after } { 4 }
            {#1} {#3} {#4} { \@contentsline@destination }
          \UseHookWithArguments { contentsline / page / before } { 4 }
            {#1} {#3} {#4}
            { \@contentsline@destination }
         \UseHookWithArguments { contentsline / page / after } { 4 }
            {#1} {#3} {#4}
            { \@contentsline@destination }
          \par
        \group_end:
        \vfil
      }
  }
```

(*End of definition for* `\l@section` *and others. These functions are documented on page* **??**.)

```
\setcounter { tocdepth } { 2 }
```

1.4 Block environments

`description` (*env.*)
`quote` (*env.*)
`quotation` (*env.*)
`verse` (*env.*)
`stdquote` (*env.*)
`stdquotation` (*env.*)
`stdverse` (*env.*)

Stub logical environments: needed as the tagging setup expects these to exist.

```
\NewDocumentEnvironment { description } { } { } { }
\NewDocumentEnvironment { quote } { } { } { }
\NewDocumentEnvironment { quotation } { } { } { }
\NewDocumentEnvironment { verse } { } { } { }
```

```
\AddToHook { begindocument / before }
  {
    \clist_map_inline:nn { quote , quotation , verse }
      {
        \NewEnvironmentCopy { std #1 } {#1}
        \RenewDocumentEnvironment {#1} { D <> { all } !O { } }
          {
            \__talk_action_begin:n {##1}
            \begin { std #1 } [ {##2} ]
              \ignorespaces
          }
          {
            \end { std #1 }
            \__talk_action_end:
          }
      }
  }
```

`block` (*env.*)

```
\NewDocumentEnvironment { block } { D <> { all } m }
  {
    \__talk_action_begin:n {#1}
    \par
    \vbox_set:Nw \l__talk_tmp_box
      \group_begin:
        \medskip
        \leavevmode
        \normalfont \large \bfseries
        \color { structure }
        #2
        \par
        \medskip
      \group_end:
  }
  {
    \vbox_set_end:
    \box_use:N \l__talk_tmp_box
    \par
    \__talk_action_end:
  }
```

1.5 Lists

`\item`
`\__talk_item_parse_spec:w`
`\__talk_item_parse_spec:n`

Again, add the additional argument: here, we have to do a little gymnastics. The test for an overlay has to come after the standard item definition: in a list, items have to *close* the structure before them first, so if we test too early, we'd end up covering then uncovering straight away!

```
\AddToHook { begindocument / before }
  {
    \NewCommandCopy \stditem \item
    \RenewDocumentCommand \item { d <> o }
      {
        \IfNoValueTF {#2}
```

```
          { \stditem }
          { \stditem [ {#2} ] }
        \IfNoValueTF {#1}
          {
            \exp_after:wN \__talk_item_parse_spec:w
              \l__talk_action_spec_str < all > \q_stop
          }
          { \__talk_item_parse_spec:n {#1} }
      }
  }
```

Parsing the spec is a separate function here as there are a couple of routes to get here. At present we only have a `false` branch, but for spacing we likely will need to add something to the `true` branch too. The odd stuff with `\currentgrouplevel` here is needed so we only close the item at the correct nesting, allowing for the group that gets added.

```
\cs_new_protected:Npn \__talk_item_parse_spec:w #1 < #2 > #3 \q_stop
  { \__talk_item_parse_spec:n {#2} }
\cs_new_protected:Npn \__talk_item_parse_spec:n #1
  {
    \tl_if_blank:nF {#1}
      {
        \tl_set:Ne \l__talk_list_end_tl
          {
            \exp_not:N \int_compare:nNnT \tex_currentgrouplevel:D =
              { \int_use:N \tex_currentgrouplevel:D + 1 }
              {
                \__talk_action_end:
                \tl_clear:N \exp_not:N \l__talk_list_end_tl
              }
          }
        \__talk_action_begin:n {#1}
      }
  }
```

(*End of definition for* `\item`*,* `\__talk_item_parse_spec:w`*, and* `\__talk_item_parse_spec:n`*. This function is documented on page* **??**.)

`\l__talk_list_end_tl`

```
\tl_new:N \l__talk_list_end_tl
```

(*End of definition for* `\l__talk_list_end_tl`.)

`\__block_inter_item:`
`\endblockenv`

There are no currently no hooks for insertion at the end of list items, so we have to do it manually. We cannot target `\__block_list_item_end:`/`\__block_list_end:` as these change definition if tagging is suspended.

```
\cs_gset_protected:Npn \__block_inter_item:
  {
    \legacy_if:nT { @inlabel }
      { \indent \par }
   \mode_if_horizontal:T
     {
       \__block_skip_remove_last:
       \__block_skip_remove_last:
       \par
     }
```

```
    \l__talk_list_end_tl
    \__kernel_list_item_end:
    \__kernel_list_item_begin:
    \addpenalty \@itempenalty
    \addvspace \itemsep
  }
\cs_gset:Npn \endblockenv
  {
    \__block_debug_typeout:n { blockenv~common~ending \on@line }
    \bool_if:NT \l__block_level_incr_bool
      { \int_gdecr:N \g_block_nesting_depth_int }
    \legacy_if:nT { @inlabel }
      {
        \mode_leave_vertical:
        \legacy_if_gset_false:n { @inlabel }
      }
    \__block_if_list:T
      { \legacy_if:nT { @newlist } { \@noitemerr } }
    \mode_if_horizontal:TF
       {
         \__block_skip_remove_last:
         \__block_skip_remove_last:
         \par
       }
       { \@inmatherr { \end { \@currenvir } } }
    \l__talk_list_end_tl
    \__kernel_displayblock_end:
    \__block_if_list:T { \legacy_if_gset_false:n { @newlist } }
    \legacy_if:nF { @noparlist }
      {
        \__block_skip_set_to_last:N \l_tmpa_skip
        \dim_compare:nNnT \l_tmpa_skip > \c_zero_dim
          {
            \skip_vertical:n { - \l_tmpa_skip }
            \skip_vertical:n { \l_tmpa_skip + \parskip - \@outerparskip }
          }
        \addpenalty \@endparpenalty
        \addvspace \l__block_topsepadd_skip
      }
    \socket_use:n { block / endpe }
  }
```

(*End of definition for* `\__block_inter_item:` *and* `\endblockenv`*. This function is documented on page* ??.)

`itemize` (*env.*)
`enumerate` (*env.*)
`description` (*env.*)

Allow for the classical beamer syntax.

```
\AddToHook { begindocument / before }
  {
    \clist_map_inline:nn { itemize , enumerate , description }
      {
        \RenewDocumentEnvironment {#1} { = { action-spec } !o }
          {
            \IfNoValueTF {##1}
              { \UseInstance { blockenv } {#1} { } }
```

```
            { \UseInstance { blockenv } {#1} {##1} }
        }
        { \endblockenv }
    }
}
```

And add the structural color to item labels.

```
\AddToHook { begindocument / before }
  {
    \EditInstance { item } { basic }
      { label-format = \color { structure } #1 }
    \EditInstance { item } { description }
      { label-format = \normalfont \bfseries \color { structure } #1 }
  }
```

\l__talk_action_spec_str

Add an overlay key to the `block` template. Placed here, it applies *before* the `\item` starts, so we do not have to redefine the latter to do actions up-front. This also means it can apply to whatever we want it to within a block.

```
\keys_define:nn { template / block / display  }
  { action-spec .str_set:N = \l__talk_action_spec_str }
```

(*End of definition for* `\l__talk_action_spec_str`.)

1.6 Theorems, *etc.*

\newtheorem
\stdnewtheorem

We need to extend the creation of theorems in two ways: add the overlay argument, and add the counter to the list of those reset during overlay creation.

```
\NewCommandCopy \stdnewtheorem \newtheorem
\RenewDocumentCommand \newtheorem { m O {#1} m o }
  {
    \IfNoValueTF {#4}
      { \stdnewtheorem {#1} [ {#2} ] {#3} }
      { \stdnewtheorem {#1} [ {#2} ] {#3} [ {#4} ] }
    \NewEnvironmentCopy { std #1 } {#1}
    \RenewDocumentEnvironment {#1} { D <> { all } o }
      {
        \__talk_action_begin:n {##1}
        \IfNoValueTF {##2}
          { \begin { std #1 } }
          { \begin { std #1 } [ {##2} ] }
            \ignorespaces
      }
      {
        \end { std #1 }
        \__talk_action_end:
      }
  }
```

(*End of definition for* `\newtheorem` *and* `\stdnewtheorem`. *These functions are documented on page* **??**.)

⟨/class⟩

Part X
ltx-talk-title – Title pages

1 ltx-talk-title implementation

Start the DocStrip guards.

```
1 ⟨*class⟩
```

Identify the internal prefix.

```
2 ⟨@@=talk⟩
```

\@author
\@date
\@institute
\@subtitle
\@title
\@shortauthor
\@shortdate
\@shortinstitute
\@shortsubtitle
\@shorttitle

We create a set of keys and variables in one go. Following the classical kernel approach, all of the underlying storage is global. The short values will always be set in the following code so can be used automatically anywhere we might want them.

```
3 \clist_map_inline:nn
4   { author , date , institute , subtitle , title }
5   {
6     \keys_define:nn { talk / metadata }
7       {
8         #1 .tl_gset:c = @ #1 ,
9         short- #1 .tl_gset:c = @short #1
10      }
11  }
```

Allow empty values for author and title.

```
12 \tl_gclear:N \@author
13 \tl_gclear:N \@title
```

As the date has a standard value, that has to be propagated.

```
14 \tl_gset_eq:NN \@shortdate \@date
```

(*End of definition for* `\@author` *and others. These variables are documented on page* **??**.)

\author
\date
\title

Slightly repetitive but as we need to handle the tagging aspects, this is easier than using a loop. The main aim is to add the short metadata concept. Notice that keys are set before the main data storage in case someone set the value as a key as well as a mandatory argument.

```
15 \RenewDocumentCommand \author { = { short-author } O { {#2} } m }
16   {
17     \keys_set:nn { talk / metadata } {#1}
18     \tl_gset:Nn \@author {#2}
19     \tl_gset_eq:NN \g__tag_title_author_tl \@author
20     \keys_set_known:nn { hyp } {#1}
21   }
22 \RenewDocumentCommand \date { = { short-date } O { {#2} } m }
23   {
24     \keys_set:nn { talk / metadata } {#1}
25     \tl_gset:Nn \@date {#2}
26   }
27 \RenewDocumentCommand \title { = { short-title } O { {#2} } m }
28   {
29     \keys_set:nn { talk / metadata } {#1}
```

```
    \tl_gset:Nn \@title {#2}
    \tl_gset_eq:NN \g__tag_title_title_tl \@title
    \keys_set_known:nn { hyp } {#1}
  }
```

(*End of definition for* `\author`, `\date`, *and* `\title`. *These functions are documented on page* ??.)

`\institute` `\subtitle`

Simple storage at present: unlike some of the kernel data, there is not a lot to do here.

```
\NewDocumentCommand \institute { = { short-institute } O { {#2} } m }
  {
    \keys_set:nn { talk / metadata } {#1}
    \tl_gset:Nn \@institute {#2}
  }
\NewDocumentCommand \subtitle { = { short-subtitle } O { {#2} } m }
  {
    \keys_set:nn { talk / metadata } {#1}
    \tl_gset:Nn \@subtitle {#2}
  }
```

(*End of definition for* `\institute` *and* `\subtitle`. *These functions are documented on page* ??.)

`\l__talk_titlelem_after_skip` `\l__talk_titlelem_before_skip` `\l__talk_titlelem_color_tl` `\l__talk_titlelem_font_tl` `\l__talk_titlelem_tag_begin_tl` `\l__talk_titlelem_tag_end_tl`

As the various elements of the titlepage share certain characteristics, we use a single template and split them as instances.

```
\NewTemplateType { titlepage-element } { 1 }
\DeclareTemplateInterface { titlepage-element } { talk } { 1 }
  {
    after-skip : length = 0em ,
    before-skip : length = 0em ,
    color : tokenlist = . ,
    font : tokenlist = \normalfont ,
    tag-begin : tokenlist = ,
    tag-end : tokenlist =
  }
\DeclareTemplateCode { titlepage-element } { talk } { 1 }
  {
    after-skip  = \l__talk_titlelem_after_skip ,
    before-skip = \l__talk_titlelem_before_skip ,
    color       = \l__talk_titlelem_color_tl ,
    font        = \l__talk_titlelem_font_tl ,
    tag-begin   = \l__talk_titlelem_tag_begin_tl ,
    tag-end     = \l__talk_titlelem_tag_end_tl
  }
  {
    \tl_if_empty:nF {#1}
      {
        \vspace { \l__talk_titlelem_before_skip }
        \group_begin:
          \tl_if_empty:NF \l__talk_titlelem_color_tl
            { \color_select:V \l__talk_titlelem_color_tl }
          \l__talk_titlelem_font_tl
          \l__talk_titlelem_tag_begin_tl
          #1
          \par
          \l__talk_titlelem_tag_end_tl
```

```
        \group_end:
        \vspace { \l__talk_titlelem_after_skip }
      }
  }
```

Standard settings are taken from beamer with minor adjustments.

```
\DeclareInstance { titlepage-element } { author } { talk }
  { before-skip = 1em }
\DeclareInstance { titlepage-element } { date } { talk }
  { after-skip = 0.5em }
\DeclareInstance { titlepage-element } { institute } { talk }
  { font = \scriptsize }
\DeclareInstance { titlepage-element } { subtitle } { talk }
  { before-skip = 0.25em , color = structure }
\DeclareInstance { titlepage-element } { title } { talk }
  {
    color = structure ,
    font = \Large ,
    tag-begin = \tag_struct_begin:n { tag = Title } ,
    tag-end = \tag_struct_end:
  }
```

(*End of definition for* `\l__talk_titlelem_after_skip` *and others.*)

`\l__talk_titlepage_order_clist`
`\l__talk_titlepage_alignment_tl`
`\l__talk_titlepage_framestyle_tl`
`\l__talk_frame_alignment_tl`

Here, we deal with the overall style: notice that frame vertical alignment actually applies elsewhere, which is why it doesn't show up in the template code part. As a result, we have a slightly repetitive key interface.

```
\NewTemplateType { titlepage } { 0 }
\DeclareTemplateInterface { titlepage } { talk } { 0 }
  {
    element-order : commalist =
      {
        title     ,
        subtitle  ,
        author    ,
        institute ,
        date
      } ,
    framestyle : tokenlist = talk ,
    horizontal-alignment : choice { left , center ,  right } = center ,
    vertical-alignment : choice { bottom , center , stretch , top } = center
  }
\DeclareTemplateCode { titlepage } { talk } { 0 }
  {
    element-order = \l__talk_titlepage_order_clist ,
    framestyle = \l__talk_titlepage_framestyle_tl ,
    horizontal-alignment =
      {
        left = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushleft } ,
        center = \tl_set:Nn \l__talk_titlepage_alignment_tl { center } ,
        right = \tl_set:Nn \l__talk_titlepage_alignment_tl { flushright }
      } ,
    vertical-alignment =
      {
        bottom = \tl_set:Nn \l__talk_frame_alignment_tl { bottom } ,
```

```
        center = \tl_set:Nn \l__talk_frame_alignment_tl { center } ,
        stretch = \tl_set:Nn \l__talk_frame_alignment_tl { stretch } ,
        top = \tl_set:Nn \l__talk_frame_alignment_tl { top }
      }
  }
  {
    \tl_if_empty:NF \l__talk_titlepage_framestyle_tl
      { \exp_args:NV \thispagestyle \l__talk_titlepage_framestyle_tl }
    \begin { \l__talk_titlepage_alignment_tl }
      \cs_set_protected:Npn \and { \quad }
      \clist_map_inline:Nn \l__talk_titlepage_order_clist
        {
          \ExpandArgs { nnv } \UseInstance { titlepage-element }
            {##1} { @ ##1 }
        }
    \end { \l__talk_titlepage_alignment_tl }
  }
```

(*End of definition for* `\l__talk_titlepage_order_clist` *and others.*)

`\maketitle` A very simple setup.

```
\NewDocumentCommand \maketitle { O {} }
  {
    \bool_if:NTF \l__talk_frame_bool
      { \UseTemplate { titlepage } { talk } {#1} }
      {
        \begin { frame }
          \UseTemplate { titlepage } { talk } {#1}
        \end { frame }
      }
  }
```

(*End of definition for* `\maketitle`. *This function is documented on page* **??**.)

⟨/class⟩

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

@ commands:
\@_decode_overlay_+:nw 121
\\ 279

A

\abovecaptionskip 139, 141
\action 102
actionenv (env.) 102
\addcontentsline 112
\addpenalty 326, 359
\AddToHook 54, 66, 143, 151, 240, 258, 278, 323, 364, 377, 393
\addtolength 48
\addvspace 327, 360
\alert 79
alertenv (env.) 89
\alt 141
\and 131
\arabic 390
\arraycolsep 25
\arrayrulewidth 27
\AssignTaggingSocketPlug 141
\author 15

B

\begin . . . 111, 112, 130, 144, 248, 397, 398
\begingroup 144, 146, 341
\belowcaptionskip 140, 142
\bfseries 21, 39, 167, 265, 382
\bigskipamount 18
block (env.) 257
block commands:
\g_block_nesting_depth_int 333
block internal commands:
__block_debug_typeout:n 331
__block_if_list:TF 339, 350
__block_inter_item: 313, 313
\l__block_level_incr_bool 332
__block_list_end: *53*
__block_list_item_end: *53*
__block_skip_remove_last: 319, 320, 343, 344
__block_skip_set_to_last:N 353
\l__block_topsepadd_skip 360
bool commands:
\bool_do_while:Nn 27
\bool_gset_false:N . . . 30, 36, 40, 414
\bool_gset_true:N . 203, 211, 217, 406
\bool_if:NTF 6, 20, 26, 32, 38, 43, 44, 46, 55, 85, 123, 125, 132, 141, 178, 249, 332, 420
\bool_lazy_and:nnTF 39, 214, 267
\bool_lazy_or:nnTF 5, 19
\bool_new:N 3, 3, 7, 8, 13, 99, 101, 384, 385, 386
\bool_set_eq:NN 117, 121
\bool_set_false:N 25, 26, 91, 110, 123, 426, 446
\bool_set_true:N 22, 27, 42, 59, 141, 178, 198, 213, 399, 434, 454
\c_false_bool 15
\c_true_bool 14
box commands:
\box_dp:N 25
\box_ht:N 60, 225, 250
\box_move_down:nn 58
\box_new:N 4, 56
\box_use:N 274
\box_use_drop:N 24, 203, 230, 234, 236, 319
\box_wd:N 30
box internal commands:
__box_dim_eval:n 22, 25, 30, 33
__box_set_to_wd: 29, 34

C

\clearpage 92
clist commands:
\clist_const:Nn 48
\clist_if_in:NnTF 55, 177
\clist_map_break: 199, 218
\clist_map_inline:Nn . . . 132, 180, 306
\clist_map_inline:nn 3, 59, 79, 121, 242, 366
\clist_new:N 10, 14
\clist_pop:NNTF 302
\clist_set:Nn 45, 176
\color 4, 11, 56, 167, 174, 184, 266, 380, 382
color commands:
\color_group_begin: 23, 35
\color_group_end: 23
\color_math:nn 9, 26
\color_math:nnn 10, 27
\color_select:n 7, 16, 21, 22, 35, 37, 48, 69, 204, 247, 300, 341

\color_select:nn 8, 17, 38
\colorlet 64
column (env.) 69
columns (env.) 9
\columnwidth 16, 75, 111
cs commands:
\cs_generate_variant:Nn 7, 8, 9, 10, 45, 46, 48, 49, 50, 51, 52, 53, 54, 175
\cs_gset:Npn 78, 79, 80, 329
\cs_gset_eq:NN 56
\cs_gset_nopar:Npn 49, 57, 64
\cs_gset_protected:Npn 18, 27, 37, 142, 142, 274, 313
\cs_if_exist:NTF 67, 75, 108, 220
\cs_if_exist_p:N 268
\cs_if_exist_use:NTF 132, 134
\cs_new:Npn . 6, 7, 34, 35, 36, 37, 38, 39, 40, 41, 42, 144, 209, 329, 389, 390
\cs_new_eq:NN 5, 6, 141, 388
\cs_new_nopar:Npn 3, 372
\cs_new_protected:Npe 53, 68, 94
\cs_new_protected:Npn 9, 11, 16, 17, 18, 24, 30, 33, 35, 36, 41, 44, 50, 51, 52, 52, 53, 55, 56, 67, 72, 81, 81, 94, 100, 101, 106, 112, 113, 116, 121, 129, 130, 130, 131, 136, 139, 144, 146, 149, 154, 157, 166, 168, 168, 168, 170, 174, 176, 178, 184, 189, 190, 196, 201, 212, 239, 265, 294, 296, 307, 335, 339, 396, 402, 410
\cs_set:Npn 14, 15, 59
\cs_set_eq:NN 142, 354, 355, 369, 370, 380, 381
\cs_set_nopar:Npn 113, 347, 349, 353, 357, 359, 364, 374, 379
\cs_set_protected:Npn 29, 98, 131, 143, 194, 206
\cs_to_str:N 297, 302
\csname 150, 153, 154

D

\date 15
\day 20
\DeclareColor 61, 67, 68, 69
\DeclareInstance 43, 52, 79, 81, 83, 85, 87, 115, 211, 212, 213, 214, 215, 216, 257, 322
\DeclareInstanceCopy 260, 325
\DeclareTemplateCode 23, 49, 54, 99, 109, 191, 228, 275
\DeclareTemplateInterface 16, 45, 47, 94, 95, 184, 218, 265
\definecolor 65
description (env.) 236, 364
dim commands:
\dim_compare:nNnTF 49, 223, 354
\dim_compare_p:nNn 50, 270
\dim_const:Nn 95, 101
\dim_eval:n 40, 41, 42
\dim_max:nn 51, 249
\dim_set:Nn 15, 74
\dim_set_eq:NN 16, 75
\dim_to_decimal:n 88
\dim_use:N 116, 117
\c_zero_dim 354
\DocumentMetadata *6*
\doublerulesep 28

E

\EditInstance 261, 326, 379, 381
\emph 282
\end 118, 119, 137, 146, 252, 347, 402
\endblockenv 313, 374
\endcsname 150, 153, 154
\endfloatenv 116, 129
\endgroup 152, 158, 354
enumerate (env.) 364
environments:
actionenv 102
alertenv 89
block 257
column 69
columns 9
description 236, 364
enumerate 364
figure 121
invisibleenv 71
itemize 364
onlyenv 71
overlayarea 197
overprint 214
quotation 236
quote 236
stdquotation 236
stdquote 236
stdverse 236
table 121
uncoverenv 71
verse 236
visibleenv 89
exp commands:
\exp_after:wN 198, 288
\exp_args:Ne 44, 129, 188
\exp_args:Nne 436
\exp_args:No 30
\exp_args:NV 129
\exp_args_generate:n 47

\exp_not:N 55, 57, 58, 59, 62, 63, 65, 68, 68, 71, 73, 74, 76, 76, 79, 85, 87, 89, 90, 91, 92, 93, 95, 96, 97, 98, 99, 101, 101, 102, 102, 103, 104, 106, 107, 108, 109, 111, 112, 112, 114, 115, 117, 118, 137, 256, 268, 301, 302, 302, 303, 306, 324
\exp_not:n 258, 326, 327, 437
\exp_stop_f: 40, 41, 42
\expandafter 153
\ExpandArgs 61, 71, 81, 89, 134, 251, 297, 298, 304, 311

F

\fboxrule 32
\fboxsep 31
\fi 18, 148
figure (env.) 121
\figurename 132
file commands:
\file_if_exist_input:nTF 88
\file_input:n 90
\footins 30
\footnotesize 185
\footskip 289, 290
fp commands:
\fp_eval:n 93
\fp_to_dim:n 103
\frame *26*, 26, 395
frame 420
frame* 420
\framesubtitle 10
\frametitle 5, 427, 437

G

\gdef 256
\geometry *3*
group commands:
\group_begin: 11, 33, 35, 58, 60, 64, 67, 84, 104, 115, 162, 202, 206, 216, 245, 262, 340
\c_group_begin_token 32
\group_end: 40, 40, 54, 62, 63, 69, 75, 87, 108, 139, 164, 207, 209, 231, 255, 270, 344
\group_insert_after:N ... 34, 159, 161

H

hbox commands:
\hbox:n 56
\hbox_set_end: 23
\hbox_set_to_wd:Nnw 14
\headsep 223, 242, 243
\hfil 17, 22, 88, 273, 317, 351, 362, 367, 377
hook commands:
\hook_gput_code:nnn 46, 92, 262
\hook_new:n 124
\hook_use:n 122
\hspace 201, 208
\hss 158
\hypersetup 146

I

\IfBooleanT 325
\ifcase 5
\IfFormatAtLeastF 7
\IfNoValueF 326, 327
\IfNoValueTF 15, 25, 36, 47, 63, 185, 283, 286, 370, 389, 396
\ignorespaces 18, 19, 80, 150, 195, 249, 399
\immediate 154
\includegraphics 316
\indent 316
\insertsection 81
\insertsubsection 81
\insertsubsubsection 81
\institute 34
int commands:
\int_compare:nNnTF 114, 191, 196, 202, 203, 208, 210, 214, 302
\int_compare_p:nNn 215, 216
\int_eval:n 188
\int_gdecr:N 333
\int_gincr:N 29, 126, 186, 398
\int_gset:Nn 187, 405
\int_gset_eq:NN 129, 134, 139
\int_gzero:N 25, 74
\int_incr:N 218
\int_max:nn 171
\int_new:N . 4, 5, 71, 128, 145, 208, 387
\int_to_Roman:n 211
\int_to_roman:n 212
\int_use:N .. 8, 14, 52, 56, 68, 303, 392
\c_max_int 190, 216
\invisible 59
invisibleenv (env.) 71
iow commands:
\iow_now:Nn 254
\item *55*, 278
itemize (env.) 364
\itemsep 54, 62, 69, 327

J

\jobname 147, 154

K

kernel internal commands:
__kernel_displayblock_end: 349

__kernel_list_item_begin: 325
__kernel_list_item_end: 324
keys commands:
\l_keys_choice_tl 71
\keys_define:nn .....
..... 3, 5, 6, 28, 58, 73, 149, 384
\keys_set:nn .. 7, 13, 17, 24, 29, 36,
41, 46, 72, 80, 162, 425, 433, 445, 453
\keys_set_known:nn 20, 32
\l_keys_value_tl 43, 159

L

\label 332
\labelenumi 34
\labelenumii 35
\labelenumiii 36
\labelenumiv 37
\labelitemfont 38, 39, 40, 41, 42
\labelitemi 38
\labelitemii 39
\labelitemiii 40
\labelitemiv 41
\labelsep 29, 33, 46, 48
\labelwidth 47, 48
\Large 21, 90
\large 265
\leavevmode 77, 264
\leftmargin 51, 59, 66
\leftmargini 43, 47, 51
\leftmarginii 44, 59
\leftmarginiii 45, 66
\leftskip 173, 183
legacy commands:
\legacy_if:nTF .....
..... 149, 252, 315, 334, 340, 351
\legacy_if_gset_false:n ... 337, 350

M

\makeatletter 145
\maketitle 139
\mathcolor 5, 11
\mbox 310
\medskip 263, 269
\mode *33*, 11
mode commands:
\mode_if_horizontal:TF 317, 341
\mode_if_math:TF 309
\mode_leave_vertical: ... 34, 312, 336
\month 5
msg commands:
\msg_error:nnn 116, 161
\msg_fatal:nn 15
\g_msg_module_name_prop 5
\g_msg_module_type_prop 6
\msg_new:nnn 277
\msg_new:nnnn 9, 222
\msg_warning:nn 273

N

\NeedsDocumentMetadata 17
\NewCommandCopy .....
..... 4, 5, 6, 280, 297, 317, 386, 395
\newcounter *19*, 72, 73, 74, 132
\NewDocumentCommand .....
..... 5, 10, 11, 34, 39, 61, 61, 81,
87, 92, 102, 139, 141, 147, 160, 173, 183
\NewDocumentEnvironment .....
..... 9, 69, 71, 89, 110, 123, 197,
214, 236, 237, 238, 239, 257, 430, 450
\NewEnvironmentCopy 244, 392
\newlabel 345
\newlength 139, 140
\NewTaggingSocket 127
\NewTaggingSocketPlug 128
\NewTemplateType .....
.... 15, 44, 46, 93, 94, 183, 217, 264
\newtheorem 386
\newwrite 153
\nobreakspace 137
\noindent 31, 157, 217, 239, 286
\normalfont 42, 50, 222, 265, 382
\normalsize 149
\number 20, 22
\numberline 117

O

\obeyedline *16*, 59
\only *40*, 59, 94
onlyenv (env.) 71
\onslide 147
opacity commands:
\opacity_select:n 17,
27, 28, 33, 34, 51, 56, 155, 158, 170, 207
\openout 154
\or .. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
overlayarea (env.) 197
overprint (env.) 214

P

\pagecolor 43
\pagestyle 383
\paperheight 57, 58
\paperwidth 58, 294, 342, 343
\par *28*, *30*, 12, 23, 25, 39, 71,
73, 83, 89, 145, 151, 192, 206, 230,
237, 260, 268, 275, 316, 321, 345, 416
\parindent 156
\parsep 53, 61, 62, 68, 69

\parskip 357
\partopsep 71
\pause 183
prg commands:
\prg_generate_conditional_variant:Nnn 10
\prg_new_protected_conditional:Npnn 3, 3
\prg_return_false: 8, 9
\prg_return_true: 7, 8
\ProcessedArgument 14, 15
\ProcessKeyOptions 87
prop commands:
\prop_gput:Nnn 5, 6
property commands:
\property_new:nnnn 8, 391
\property_record:nn 52, 68, 394
\property_ref:nn 14
\protect 117
\ProvidesExplClass 3
\put 57

Q

\quad 131
quark commands:
\quark_if_recursion_tail_stop:N 131
\quark_if_recursion_tail_stop_do:Nn 148, 159
\quark_if_recursion_tail_stop_do:nn 37
\q_recursion_stop 34, 124, 151
\q_recursion_tail 34, 124, 151
\q_stop 63, 71, 97, 99, 102, 108, 181, 184, 199, 201, 289, 294
quotation (env.) 236
quote (env.) 236

R

\raggedright 78, 129, 194
\refstepcounter 95
\relax 154
\relsize 93
\RenewCommandCopy 26
\RenewDocumentCommand . 11, 15, 21, 22, 27, 30, 43, 94, 281, 298, 318, 332, 387
\RenewDocumentEnvironment 245, 368, 393, 422, 442
\RequirePackage 3, 91, 112, 130, 133, 134, 139, 140, 145, 316
\rmdefault 142
\rule 58
rule commands:
\rule:nnn 37, 342

S

scan commands:
\scan_stop: 43
\scriptsize 84
\section 72, 81
seq commands:
\seq_gput_right:Nn 146
\seq_gremove_all:Nn 75, 76, 77
\seq_map_indexed_inline:Nn 83
\seq_map_inline:Nn 126, 133, 138
\seq_new:N 118
\seq_set_from_clist:Nn 81, 119
\l_tmpa_seq 81, 83
\setcounter 24, 235
\setlength 25, 26, 27, 28, 29, 31, 32, 33, 43, 44, 45, 46, 47, 71, 141, 142
\setmathfont 136
\setsansfont 135
\SetTemplateKeys 110
\sfdefault 142
\skip 30
skip commands:
\skip_horizontal:n 244, 291, 320, 337, 343
\skip_set:Nn 173, 183
\skip_vertical:n 96, 98, 102, 104, 108, 110, 114, 116, 356, 357
\l_tmpa_skip 353, 354, 356, 357
socket commands:
\socket_use:n 362
\space 19, 21
\stdcolor 4
\stdemph 282
\stdincludegraphics 316
\stditem 280, 284, 285
\stdmathcolor 4
\stdnewtheorem 386
stdquotation (env.) 236
stdquote (env.) 236
\stdtextbf 282
\stdtextcolor 4
\stdtextit 282
\stdtextmd 282
\stdtextnormal 282
\stdtextrm 282
\stdtextsc 282
\stdtextsf 282
\stdtextsl 282
\stdtexttt 282
\stdtextup 282
stdverse (env.) 236
\stepcounter *19*
str commands:
\str_clear:N 18, 28, 29

\str_if_empty:NTF 86
\str_if_empty_p:N 40
\str_if_eq:nnTF 17, 57, 83, 108, 110, 331
\str_if_eq_p:nn 6, 7, 21
\str_new:N 9, 11, 12, 15, 100
\str_put_right:Nn 134, 170
\str_replace_all:Nnn 20, 22, 52
\str_set:Nn 18, 24, 67, 71, 111
\str_set_eq:NN 119
\string 345
\subsection 72, 81
\subsubsection 72, 81
\subtitle 34
sys commands:
\sys_if_engine_opentype:TF 131

T

\tabbingsep 29
\tabcolsep 26
table (env.) 121
\tableename 132
\tableofcontents 160
tag commands:
\tag_get:n 405
\tag_mc_begin:n 173, 180, 412
\tag_mc_end: 171, 178, 418
\tag_resume:n 170, 417
\tag_struct_begin:n . 91, 132, 172, 404
\tag_struct_end: . . . 92, 139, 179, 408
\tag_suspend:n 181, 413
tag internal commands:
\g__tag_title_author_tl 19
\g__tag_title_title_tl 31
\tagpdfparaOff 61
\tagpdfsetup 147
talk internal commands:
__talk_action_:N 17, 17
__talk_action_alert:N 18, 18
__talk_action_begin:n 11, 79, 102, 105, 111, 113, 125, 247, 259, 309, 395
__talk_action_end: *30*, 26, 84, 102, 107, 112, 130, 130, 253, 276, 305, 403
__talk_action_invisible:N . . 24, 24
__talk_action_only:N 36
__talk_action_only_begin:N 36
__talk_action_only_end:N . . . 36, 41
\l__talk_action_spec_str 149, 289, 384
__talk_action_uncover:N 53, 53
__talk_action_visible:N 24, 30
\l__talk_aspect_ratio_str . . 58, 107
\l__talk_cnt_reset_seq 75, 76, 77, 118, 133, 138, 146
__talk_cnt_restore: 86, 131, 136
__talk_cnt_save: 77, 131, 131
__talk_column_align_bottom:n 50, 50
__talk_column_align_center:n 50, 52
__talk_column_align_top:n . . 50, 67
\l__talk_column_alignment_tl . 28, 86
\l__talk_columns_wd_tl 5, 14, 15
__talk_decode_action:n . . 85, 94, 94
__talk_decode_action:w . . 94, 96, 101
\l__talk_decode_action_str 12, 18, 111, 120, 125
\l__talk_decode_actions_bool 13, 25, 122, 126
\l__talk_decode_actions_clist . . . 13
\l__talk_decode_actions_str . . 13, 29
\l__talk_decode_arg_str 9, 24, 30, 117, 162
__talk_decode_check:n . 127, 174, 174
__talk_decode_check:nw 174, 181, 184
__talk_decode_check_range:nnn 174, 190, 191, 206
__talk_decode_check_single:nn 174, 187, 194
__talk_decode_mode:n 44, 53, 53
__talk_decode_mode:nn . . . 76, 79, 81
__talk_decode_mode:w 53, 62, 68
__talk_decode_mode_aux:n 53
__talk_decode_overlay_.:nw 121
__talk_decode_overlay_aux:nNN 121, 142, 145, 146
__talk_decode_overlay_offset:nNn 121, 150, 155, 165, 168
__talk_decode_overlay_offset:nNnN 121, 154, 157, 166
__talk_decode_overlays:nN 121, 124, 129, 135, 172
__talk_decode_overlays:nn 87, 106, 113, 121, 121
\l__talk_decode_overlays_bool 3, 6, 22, 26, 42, 59, 118, 123
\l__talk_decode_overlays_clist . . 10
\l__talk_decode_overlays_str 10, 28, 40, 86
__talk_decode_parse:n . 5, 16, 16, 116
__talk_decode_parse:w . 16, 34, 35, 46
__talk_decode_parse_aux:n 16, 30, 33
\l__talk_decode_pure_bool 7, 27, 41, 91, 110
\l__talk_decode_step_bool 8, 123, 125, 141
\l__talk_float_alignment_tl 92, 104, 105, 106, 112, 118
\l__talk_fontsize_dim 58, 88, 93
\l__talk_footelem_color_tl 183
\l__talk_footelem_font_tl 183
\l__talk_footelem_left_skip 183

\l__talk_footelem_right_skip ... 183
\l__talk_footer_bg_tl 264
\l__talk_footer_fg_tl 264
\l__talk_footer_font_tl 264
\l__talk_footer_left_skip 264
\l__talk_footer_order_clist 264
\l__talk_footer_right_skip 264
\l__talk_footer_sep_tl 264
\l__talk_frame_alignment_tl ....
..... 89, 94, 148, 158
\l__talk_frame_bool 141, 384, 399
\g__talk_frame_int .....
..... 14, 52, 68, 211, 387, 392, 398
__talk_frame_notag:n ... 41, 410, 410
__talk_frame_overprint: .....
..... 209, 209, 221, 224, 228,
241, 243, 244, 247, 249, 256, 258, 263
__talk_frame_process:nn .....
..... 396, 396, 427, 436, 447, 455
\g__talk_frame_struct_int 56, 71, 405
\g__talk_frame_subtitle_tl 3, 13, 76
__talk_frame_tag:n 37, 402, 402
\g__talk_frame_tag_bool .....
..... 46, 385, 406, 414
\l__talk_frame_tagging_str ....
..... 17, 18, 20, 22, 34, 149
__talk_frame_title:n 15, 38, 44
\l__talk_frame_title_bool .. 58, 420
__talk_frame_title_tagged:n ...
..... 15, 47, 51
\g__talk_frame_title_tl .....
..... 3, 8, 58, 75, 253, 435
\l__talk_frame_verb_bool .....
..... 44, 386, 426, 434, 446, 454
\l__talk_frametitle_after_skip .
..... 25, 41
\l__talk_frametitle_before_skip
..... 26, 32
\l__talk_frametitle_color_tl ...
..... 27, 34, 35
\l__talk_frametitle_font_tl .. 28, 36
\l__talk_header_bg_tl 217
\l__talk_header_fg_tl 217
\l__talk_header_font_tl 217
\l__talk_header_frametitle_bool 217
\l__talk_header_ht_dim 217
\l__talk_header_left_skip 217
\l__talk_header_right_skip 217
__talk_header_tag_begin:n ....
..... 53, 168, 168, 175
__talk_header_tag_end: . 64, 168, 176
__talk_if_overlay:n 3, 10
__talk_if_overlay:nTF .....
..... 3, 7, 12, 13, 13, 23, 31, 32,
34, 45, 96, 143, 152, 175, 301, 320, 335
__talk_item_parse_spec:n .....
..... 278, 291, 295, 296
__talk_item_parse_spec:w .....
..... 278, 288, 294
__talk_label:n 332, 336, 339
__talk_latexe_frame:n .. 26, 395, 395
\l__talk_list_end_tl .....
..... 300, 306, 312, 323, 348
__talk_metadata_name:n .....
..... 305, 308, 313, 329, 329
__talk_mode:n 3
__talk_mode:nTF 3, 12
\l__talk_mode_str 7, 58, 58, 83
\c__talk_modes_clist 48, 55
__talk_onslide:n . 147, 148, 149, 188
\g__talk_onslide_escape_tl ....
..... 156, 159, 161, 165, 171
__talk_onslide_reset: .....
..... 147, 153, 166, 170
\g__talk_onslide_tl .....
..... 79, 83, 151, 162, 164, 171
__talk_overlay_arg:n .....
..... 3, 11, 62, 72, 82, 90
__talk_overprint_begin:n .....
..... 190, 190, 198, 215
__talk_overprint_check_ht:n ...
..... 214, 263, 265, 274
\l__talk_overprint_int . 208, 212, 218
__talk_overprint_save_ht: ....
..... 214, 219, 239
__talk_pagecolor:n 43, 48, 49, 52
\c__talk_paper_height_dim 95
\c__talk_paper_width_dim 95
\g__talk_pauses_int .....
.... 9, 4, 74, 126, 171, 186, 187, 188
\l__talk_saved_action_str 99, 119, 135
\l__talk_saved_actions_bool ....
..... 99, 121, 137
\l__talk_saved_overlays_bool ...
..... 99, 117, 132
__talk_sect_tag:nn 127, 129, 130
\g__talk_section_tl 66
\l__talk_section_tl 66
__talk_slide:nn 9, 9, 400
__talk_slide_align_bottom:n . 94, 94
__talk_slide_align_center:n 94, 100
__talk_slide_align_stretch:n 94, 106
__talk_slide_align_top:n .. 94, 112
__talk_slide_aux:n 9, 45, 56
__talk_slide_begin: 33, 72, 72
\l__talk_slide_box 4, 78, 90

\g__talk_slide_continue_bool . . 3, 27, 30, 36, 40, 85, 178, 203, 211, 217
__talk_slide_end: 49, 72, 81
\g__talk_slide_int 5, 8, 25, 29, 196, 202, 208, 210, 215
\g__talk_subsection_tl 66
\l__talk_subsection_tl 66, 109
\g__talk_subsubsection_tl 66
\l__talk_subsubsection_tl . . 66, 111
__talk_textcmd_eqiv:n . 282, 303, 307
\l__talk_titlelem_after_skip 44
\l__talk_titlelem_before_skip . . . 44
\l__talk_titlelem_color_tl 44
\l__talk_titlelem_font_tl 44
\l__talk_titlelem_tag_begin_tl . . 44
\l__talk_titlelem_tag_end_tl 44
\l__talk_titlepage_alignment_tl . 94
\l__talk_titlepage_framestyle_tl 94
\l__talk_titlepage_order_clist . . 94
__talk_tmp:w 55, 55, 98, 107
\l__talk_tmp_box 14, 24, 39, 56, 60, 73, 87, 193, 203, 225, 230, 234, 236, 250, 261, 274, 292, 319
\l__talk_tmp_tl 12, 18, 21, 23, 57, 101, 302, 304, 305
__talk_toc_aux:nnnn 166, 167, 170, 180, 189
__talk_toc_dest:n 166, 193, 196
__talk_toc_dest:w 166, 198, 201
__talk_toc_level:nnnn . 166, 194, 212
\l__talk_uncover_hidden_fp 46
__talk_wallpaper_hrule:Nnn 240, 287, 335, 335
talk/sec/title 127
\temporal 173
TEX and LATEX 2ε commands:
\@arabic 6, 7, 78, 79, 80, 144, 389
\@author 3, 18, 19
\@auxout 254, 343
\@bsphack 334
\@caption 32, 143
\@captype 113
\@contentsline@destination 51, 198, 220, 223, 226, 229
\@currentHref 350
\@currentlabel 347
\@currentlabelname 349
\@currenvir 347
\@date 3, 25
\@definecounter 141
\@endparpenalty 359
\@esphack 337
\@evenfoot 355, 370, 381
\@evenhead 354, 369, 380
\@framenumber 387
\@ignore 30
\@ignoretrue 90
\@inmatherr 347
\@input 147
\@institute 3, 37
\@itempenalty 326
\@kernel@reserved@label@data . . . 351
\@listI 56
\@listi 49, 56
\@listii 57
\@listiii 64
\@makecaption 150
\@makefnmark 158
\@makefntext 32, 154
\@mpfootins 30
\@nobreakfalse 157
\@noitemerr 340
\@oddfoot . 353, 355, 364, 370, 379, 381
\@oddhead . 349, 354, 359, 369, 374, 380
\@outerparskip 357
\@parboxrestore 76, 147
\@setminipage 148
\@shortauthor 3
\@shortdate 3
\@shortinstitute 3
\@shortsubtitle 3
\@shorttitle 3
\@starttoc 142, 163
\@subtitle 3, 42
\@title 3, 30, 31
\c@figure 132
\c@frame 387
\c@page 144
\c@pauses 4
\c@section 78
\c@slide 5
\c@subsection 79
\c@subsubsection 80
\c@table 132
\check@mathfonts 143
\currentgrouplevel 53
\fnum@figure 132
\fnum@table 132
\Gm@bmargin 290
\Gm@lmargin 224, 271, 337
\Gm@rmargin 226, 272, 320
\Gm@tmargin 223
\hb@xt@ 158
\if@minipage 148
\ignorespaces 30
\l@section 166
\l@subsection 166
\l@subsubsection 166

\on@line 331
\protected@write 343
\ps@plain 347
\ps@talk 347
\ps@wallpaper 347
\std@definecounter 141
tex commands:
\tex_currentgrouplevel:D . . 302, 303
\tex_fontdimen:D 61
\tex_hsize:D 22, 33
\tex_setbox:D 20, 31
\tex_textfont:D 61
\tex_vbox:D 20, 31
\tex_vrule:D 39
text commands:
\text_purify:n 53, 58, 129
\text_titlecase_first:n 134
\textasteriskcentered 40
\textbf 282
\textbullet 38
\textcolor 6, 11
\textendash 39
\textheight 87
\textit 282
\textmd 282
\textnormal 282
\textperiodcentered 41
\textrm 282
\textsc 282
\textsf 282
\textsl 282
\texttt 282
\textup 282
\textwidth *28*, 8, 15, 16, 74, 75, 214
\theenumi 34
\theenumii 35
\theenumiii 36
\theenumiv 37
\thefigure 132
\theframe 387
\thepage *5*, 144, 348
\thepauses 4
\thesection 72
\theslide 5
\thesubsection 72
\thesubsubsection 72
\thetable 132
\thispagestyle 129
\tiny 270
\title 15
tl commands:
\tl_clear:N 109, 111, 306
\tl_gclear:N 12, 13, 75, 76, 79, 164, 165
\tl_gset:Nn 8, 13, 18, 25, 30, 37, 42, 156, 162, 244, 247, 435
\tl_gset_eq:NN 14, 19, 31
\tl_if_blank:nTF 37, 74, 90, 105, 112, 189, 298
\tl_if_blank_p:n 20
\tl_if_empty:NTF 34, 68, 128, 203, 246, 299, 308, 338
\tl_if_empty:nTF 64, 186, 199
\tl_if_exist:NTF 241
\tl_map_inline:nn 282
\tl_new:N 3, 4, 57, 66, 67, 68, 69, 70, 71, 92, 133, 135, 148, 171, 172, 243, 312
\tl_retokenize:n 63
\tl_set:Nn 12, 104, 105, 106, 106, 115, 116, 117, 121, 122, 123, 124, 134, 136, 300
\tl_set_eq:NN 42, 158
\tl_to_str:n 50, 51, 62, 77, 97, 99, 102, 108, 437
\tl_trim_spaces:n 45
\tl_use:N 83, 89, 151
\today 3
token commands:
\token_if_eq_meaning:NNTF . 153, 164
\token_to_str:N 69, 70
\topsep 52, 60, 67

U

\uncover 59
uncoverenv (env.) 71
\unskip 21
use commands:
\use:N 86, 89, 96, 99, 103, 125, 135
\use:n 46, 48, 60, 66, 85, 96, 99, 110, 192, 322
\use_none:n 136
\UseHookWithArguments 219, 222, 224, 227, 342
\UseInstance . . 57, 126, 128, 134, 252, 304, 312, 361, 366, 371, 372, 376, 379
\UseStructureName 102, 135
\UseTaggingSocket . . 96, 97, 107, 146, 148
\UseTemplate 142, 145

V

\value 115, 191, 203, 214
vbox commands:
\vbox:n 51, 54, 63
\vbox_set:Nw 39, 78, 261
\vbox_set_end: 44, 84, 85, 200, 217, 273
\vbox_set_to_wd:Nnn 18, 292
\vbox_set_to_wd:Nnw 27, 73, 193
\vbox_to_ht:nn 54, 87, 201, 227

\vbox_top:n 68
\vbox_unpack_drop:N 87, 90
vcoffin commands:
\vcoffin_set:Nnn *1*
verse (env.) 236
\vfil 204, 231, 232
\visible 79
visibleenv (env.) 89
\vspace 32, 41, 66, 76

Y

\year 22