1 Package header

```
1 ⟨*package⟩
2 ⟨@@=yoin⟩
```

Necessary packages: First, LaTeX3 stuff.

```
3 \RequirePackage{expl3,l3keys2e,l3regex,xparse}
```

From zref bundle, for computing the total number of pages of an article.

```
4 \RequirePackage{zref-totpages}
```

We need the absolute paths. This also means we need -recorder option to pdflatex.

```
5 \RequirePackage[abspath]{currfile}
```

For including PDF files.

```
6 \RequirePackage{pdfpages}
```

Package header.

```
7 \ProvidesExplPackage{yoin}{2016/02/28}{v0.0.1}{Joining articles into issues}
```

2 General macros

Macros not necessarily related to the package; moreorless an addition to LaTeX3.

\yoin_seq_gappend_clist:Nn   Globally append clist #2 to seq #1.

```
8  \seq_new:N \l__yoin_seq_tmpa_seq
9  \cs_new_protected:Nn \yoin_seq_gappend_clist:Nn {
10     \seq_set_from_clist:Nn \l__yoin_seq_tmpa_seq { #2 }
11     \seq_gconcat:NNN #1 #1 \l__yoin_seq_tmpa_seq
12 }
```

\yoin_keys_set_from_file:nn   Read a file #2 containing a key–value list and set the keys for #1. No checks are done here, nothing like comments could be used, the keys should be separated by a comma (and spaces of course as needed).

```
13 \tl_new:N \l__yoin_keys_tmpa_tl
14 \cs_generate_variant:Nn \keys_set:nn { nV }
15 \cs_new_protected:Nn \yoin_keys_set_from_file:nn {
16     \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #2 }
17     \keys_set:nV { #1 } \l__yoin_keys_tmpa_tl
18 }
```

\yoin_keyval_parse_from_file:nn   Read a file #2 containing a key–value list and set the keys for #1. No checks are done here, nothing like comments could be used, the keys should be separated by a comma (and spaces of course as needed).

```
19 \cs_generate_variant:Nn \keyval_parse:NNn { NNV }
20 \cs_new_protected:Nn \yoin_keyval_parse_from_file:NNn {
```

```
21     \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #3 }
22     \keyval_parse:NNV #1 #2 \l__yoin_keys_tmpa_tl
23 }
```

msg:␣boolean-values-only   Message for a non-boolean passed to a bool key.

```
24 \msg_new:nnn { yoin } { boolean-values-only }
25     { Key ~ '#1' ~ accepts ~ boolean ~ values ~ only.}
```

3   Key–value interface for the package setup

First, we define the variables to store the keys.

\g_yoin_subprocess_bool   Booleans:
\g_yoin_article_bool
\g_yoin_dryrun_bool
\g_yoin_onlyflags_bool
\g_yoin_onlytags_bool

```
26 \bool_new:N \g_yoin_subprocess_bool
27 \bool_new:N \g_yoin_article_bool
28 \bool_new:N \g_yoin_dryrun_bool
29 \bool_new:N \g_yoin_onlyflags_bool
30 \bool_new:N \g_yoin_onlytags_bool
```

\g_yoin_flags_seq   Sequences for flags, tags and their filtering:
\g_yoin_tags_seq
\g_yoin_onlyflags_seq
\g_yoin_onlytags_seq

```
31 \seq_new:N \g_yoin_flags_seq
32 \seq_new:N \g_yoin_tags_seq
33 \seq_new:N \g_yoin_onlyflags_seq
34 \seq_new:N \g_yoin_onlytags_seq
```

\g_yoin_jobname_tl   We can modify what the package considers as the value of \jobname, here's a token list for that:

```
35 \tl_new:N \g_yoin_jobname_tl
36 \tl_gset_eq:NN \g_yoin_jobname_tl \c_sys_jobname_str
```

msg:␣unknown-flag   Two messages, for unknown flags and unknown tags.
msg:␣unknown-tag

```
37 \msg_new:nnnn { yoin } { unknown-flag }
38     { The ~ flag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
39     { You ~ either ~ misspelled ~ it ~ or ~forgot ~ to ~ declare ~ it. }

40 \msg_new:nnnn { yoin } { unknown-tag }
41     { The ~ tag ~ '#1' ~ is ~ unknown ~ to ~ 'yoin'. }
42     { You ~ either ~ misspelled ~ it ~ or ~forgot ~ to ~ declare ~ it. }
```

`\yoin_if_tag_defined:n`
`\yoin_if_flag_defined:n`

Conditionals for checking whether a tag/flag was defined.

```
43 \prg_new_protected_conditional:Nnn \yoin_if_tag_defined:n { T, F, TF } {
44     \seq_if_in:NnTF \g_yoin_tags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
45 }
```

```
46 \prg_new_protected_conditional:Nnn \yoin_if_flag_defined:n { T, F, TF } {
47     \seq_if_in:NnTF \g_yoin_flags_seq { #1 } { \prg_return_true: } { \prg_return_false: }
48 }
```

`\__yoin_error_if_tag_undefined:n`
`\__yoin_error_if_flag_undefined:n`

Check whether a tag/flag is defined, if not, issue an error.

```
49 \cs_new_protected:Nn \__yoin_error_if_tag_undefined:n {
50     \yoin_if_tag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-tag } { #1 } }
51 }
```

```
52 \cs_new_protected:Nn \__yoin_error_if_flag_undefined:n {
53     \yoin_if_flag_defined:nF { #1 } { \msg_error:nnn { yoin } { unknown-flag } { #1 } }
54 }
```

yoin␣/␣general

The keys themselves:

```
55 \keys_define:nn { yoin / general } {
```
Booleans:
```
56     dryrun .bool_gset:N = \g_yoin_dryrun_bool,
57     dryrun .initial:n = { false },
```

```
58     article .bool_gset:N = \g_yoin_article_bool,
59     article .initial:n = { false },
```

```
60     subprocess .bool_gset:N = \g_yoin_subprocess_bool,
61     subprocess .initial:n = { false },
```
Keys whose `clist` values are appended to a seq:
```
62     defineflags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_flags_seq { #1 },
```

```
63     definetags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_tags_seq { #1 },
```
A `clist` key is stored in a `seq`, also, a corresponding `bool` is set true. (The point is, if `onlyflags`/`onlytags` is not ever set up, we want to know it since we treat it as if we use all flags/tags.)
```
64     onlyflags .code:n =
65         \seq_gset_from_clist:Nn \g_yoin_onlyflags_seq { #1 }
66         \bool_gset_true:N \g_yoin_onlyflags_bool
67         ,
```

```
68    onlytags .code:n =
69        \seq_gset_from_clist:Nn \g_yoin_onlytags_seq { #1 }
70        \bool_gset_true:N \g_yoin_onlytags_bool
71            ,
```

A key whose value is stored in a token list.

```
72    jobname .tl_gset:N = \g_yoin_jobname_tl,

73 % A key that allows |\yoinMeta| to be called from within the package options.
74 %    \begin{macrocode}
75    meta .code:n = \yoin_yoinmeta:n { #1 },

76 }
```

`\yoin_setup:n` Allow keys to be set later. We define both a LaTeX3 interface and an xparse UI wrapper.
`\yoinSetup`
```
77 \cs_new_protected:Nn \yoin_setup:n {
78    \keys_set:nn { yoin / general } { #1 }
79 }

80 \NewDocumentCommand \yoinSetup { R[]{} } {
81    \yoin_setup:n { #1 }
82 }
```

4   yoinMeta macro — adding issue's metadata

`\yoinMeta`
`\yoin_yoinmeta:n`
```
83 \prop_new:N \l__yoin_yoinmeta_prop
84 \cs_new_protected:Nn \__yoin_yoinmeta_storekey:nn {
85    \prop_put:Nnn \l__yoin_yoinmeta_prop { #1 } { #2 }
86 }
87 \cs_new_protected:Nn \__yoin_yoinmeta_storekey:n {
88    \prop_put:Nnn \l__yoin_yoinmeta_prop { #1 } { }
89 }
90 \cs_new_protected:Nn \yoin_yoinmeta:n {
91    \keyval_parse:NNn \__yoin_yoinmeta_storekey:n \__yoin_yoinmeta_storekey:nn { #1 }
92 }
93 \NewDocumentCommand \yoinMeta { R[]{} } {
94    \yoin_yoinmeta:n { #1 }
95 }
```

5 yoinAdd macro — adding articles to the issue

The key–value interface. In this case, we basically only store the keys for each article in a prop. First, an interface for setting the keys for the articles. `\yoin_yoinadd_prop:n` returns the name of the prop for the given article; *no check for existence is done at this place.*

`\g_yoin_yoinadd_seq` A sequence for storing the list of the existing articles.

```
96 \seq_new:N \g_yoin_yoinadd_seq
```

`\yoin_yoinadd_prop:n`
`\yoin_yoinadd_prop:V`
`\yoin_yoinadd_prop_item:nn`
`\yoin_yoinadd_prop_item:Vn`

`\yoin_yoinadd_prop:n` returns the name of the prop for the given article; *no check for existence is done at this place.* `\yoin_yoinadd_prop:nn` returns property `\#2` of article `\#1`, or `\q_no_value` if the property is not set.

```
97 \cs_new:Nn \yoin_yoinadd_prop:n {
98     g__yoin_article_#1_prop
99 }
100 \cs_generate_variant:Nn \yoin_yoinadd_prop:n { V }

101 \cs_new:Nn \yoin_yoinadd_prop_item:nn {
102     \prop_item:cn { \yoin_yoinadd_prop:n { #1 } } { #2 }
103 }
104 \cs_generate_variant:Nn \yoin_yoinadd_prop_item:nn { V }
```

For processing `\yoinAdd`, we first set up a `tl` to contain the name of the article, then create the prop, and finally use `l3keys` to fill in the prop. Note that if an article is added twice, an error is issued, if the error is ignored, the article is not added but the properties are set.

`\l__yoin_yoinadd_currentarticle_tl` A `tl` that stores the name of the article that is being processed by `\yoinAdd`.

```
105 \tl_new:N \l__yoin_yoinadd_currentarticle_tl
```

`\__yoin_yoinadd_storekey:nn`
`\__yoin_yoinadd_storekey:n`

Internal macro for storing a key in the prop. The one-parameter variant sets the value of the key empty.

```
106 \cs_new_protected:Nn \__yoin_yoinadd_storekey:nn {
107     \prop_gput:cnn { \yoin_yoinadd_prop:V \l__yoin_yoinadd_currentarticle_tl } { #1 } { #2 }
108 }
109 \cs_new_protected:Nn \__yoin_yoinadd_storekey:n {
110     \prop_gput:cnn { \yoin_yoinadd_prop:V \l__yoin_yoinadd_currentarticle_tl } { #1 } { }
111 }
```

`\yoin_yoinadd:nn`
`\yoinAdd`

The macro `\yoinAdd` itself. We first set `\l_@@_yoinadd_currentarticle_tl`, then check whether the same article has not been processed before (issuing an error in that case and finishing). Then, the article is added in `\g_yoin_yoinadd_seq`, the prop created, the article's name added in the prop with key `article` and the keys are set. If the article has a `.yoin` file in its sub-directory, the

key–values in it is added to the prop. If the file does not exist, it means things are wrong (the article should first be set up, before being added to its issue by \yoinAdd).

```
112 \cs_new_protected:Nn \yoin_yoinadd:nn {
113     \tl_set:Nn \l__yoin_yoinadd_currentarticle_tl { #1 }
114     \seq_if_in:NnTF \g_yoin_yoinadd_seq { #1 } {
115         \msg_error:nnn { yoin } { yoinadd-duplicatearticle } { #1 }
116     } {
117         \seq_gput_right:Nn \g_yoin_yoinadd_seq { #1 }
118         \prop_new:c { \yoin_yoinadd_prop:n { #1 } }
119         \clist_map_inline:nn { forceopenany, forceopenright, ignore } {
120             \__yoin_yoinadd_storekey:nn { ##1 } { 0 }
121         }
122         \__yoin_yoinadd_storekey:nn { article } { #1 }
123         \keys_set:nn { yoin / yoinadd } { #2 }
124         \file_if_exist:nTF { #1 / #1 .yoin } {
125             \yoin_keyval_parse_from_file:NNn
126                 \__yoin_yoinadd_storekey:n
127                 \__yoin_yoinadd_storekey:nn
128                 { #1 / #1 .yoin }
129         } {
130             \msg_error:nnn { yoin } { yoinadd-dotyoinmissing } { #1 }
131         }
132     }
133 }

134 \NewDocumentCommand \yoinAdd { m O{} } {
135     \yoin_yoinadd:nn { #1 } { #2 }
136 }
```

msg:␣yoinadd-duplicatearticle
msg:␣yoinadd-dotyoinmissing
The error messages: for adding a duplicate article and for adding an article with no #1/#1.yoin file.

```
137 \msg_new:nnn { yoin } { yoinadd-duplicatearticle }
138     { The ~ article ~ "#1" ~ has ~ been ~ already ~ processed ~ by ~ \token_to_str:N \yoinAdd ~.}
139 \msg_new:nnn { yoin } { yoinadd-dotyoinmissing }
140     { The ~ article ~ "#1" ~ has ~ no ~ file "#1/#1.yoin" ~ and ~ was ~ not ~ properly ~ set ~ up.}
```

yoin␣/␣yoinadd
The keys here are pretty simple; each defined key just stores its value in the prop. We recall that \#1 is the key and \#\#1 is the value.

```
141 \clist_map_inline:nn { textualkey } {
142     \keys_define:nn { yoin / yoinadd } {
143         #1 .code:n = \__yoin_yoinadd_storekey:nn { #1 } { ##1 },
```

```
144        }
145 }
```

For boolean keys, we create a manual boolean parser.

```
146 \clist_map_inline:nn { forceopenany, forceopenright, ignore } {
147     \keys_define:nn { yoin / yoinadd } {
148         #1 .choice:,
149         #1 / true  .code:n = \__yoin_yoinadd_storekey:nn { #1 } { 1 },
150         #1 / false .code:n = \__yoin_yoinadd_storekey:nn { #1 } { 0 },
151         #1 / unknown .code:n = \msg_error:nnx { yoin } { boolean-values-only } { \l_keys_key_tl },
152     }
153 }
```

However, for the `tag` key, we additionally check that the tag exists.

```
154 \keys_define:nn { yoin / yoinadd } {
155     tag .code:n =
156         \__yoin_error_if_tag_undefined:n { #1 }
157         \__yoin_yoinadd_storekey:nn { tag } { #1 }
158         ,
159 }
```

6   Environment *yoinshell*

<code>\l_yoin_yoinshell_ignore_bool</code> A boolean for storing the `ignore` key's value.

<code>yoin␣/␣yoinshell</code>   Key–value interface to `yoinshell`.

```
160 \keys_define:nn { yoin / yoinshell } {
```

If `flag` is set and `onlyflags` is set but the flag is not amongst them, the whole `yoinshell` is ignored (by setting the `ignore` key).

```
161     flag .code:n =
162         \__yoin_error_if_flag_undefined:n { #1 }
163         \bool_if:NT \g_yoin_onlyflags_bool {
164             \seq_if_in:NnF \g_yoin_onlyflags_seq { #1 } {
165                 \keys_set:nn { yoin / yoinshell } {
166                     ignore = true
167                 }
168             }
169         }
170         ,
```

The `ignore` key sets a boolean

```
171    ignore .bool_set:N = \l_yoin_yoinshell_ignore_bool,
172    ignore .initial:n = { false },

173 }
```

`shellesc.sty` A reasonable shell escape that should work in both `pdflatex` and `lualatex` in TEX Live 2016.
`\ShellEscape`
`\__yoin_yoinshell_shellescape:n`
```
174 \file_if_exist:nTF { shellesc.sty } {
175    \RequirePackage { shellesc }
176 } {
177    \def \ShellEscape #1 { \immediate \write 18 { #1 } }
178 }
179 \cs_new_protected:Nn \__yoin_yoinshell_shellescape:n {
180    \ShellEscape { #1 }
181 }
182 \cs_generate_variant:Nn \__yoin_yoinshell_shellescape:n { V }
```

`\__yoin_yoinshell_begin:n` Environment `yoinshell` (one key–value argument). We perform some local definitions that should stay local, so we put everything
`\__yoin_yoinshell_end:` in a group. The keys are set. Then we define the macros — "shell commands". If `ignore` is set or if subprocess is passed to the
`{yoinshell}` package, these macros are declared to do nothing, otherwise they are simply wrappers to the LATEX3 counterparts.
```
183 \cs_new_protected:Nn \__yoin_yoinshell_begin:n {
184    \group_begin:
185    \keys_set:nn { yoin / yoinshell } { #1 }
186    \bool_if:NT \g_yoin_subprocess_bool {
187        \bool_set_true:N \l_yoin_yoinshell_ignore_bool
188    }
189    \bool_if:NTF \l_yoin_yoinshell_ignore_bool {
190        \DeclareDocumentCommand \RunForEach { O{} m } { }
191        \DeclareDocumentCommand \AutoRunForEach { O{} } { }
192        \DeclareDocumentCommand \Run { O{} m } { }
193        \DeclareDocumentCommand \AutoRun { O{} } { }
194        \DeclareDocumentCommand \WriteMeta { O{} } { }
195    } {
196        \DeclareDocumentCommand \RunForEach { O{} m } { \yoin_yoinshell_runforeach:nn { ##1 } { ##2 } }
197        \DeclareDocumentCommand \AutoRunForEach { O{} } { \yoin_yoinshell_autorunforeach:n { ##1 } }
198        \DeclareDocumentCommand \Run { O{} m } { \yoin_yoinshell_run:nn { ##1 } { ##2 } }
199        \DeclareDocumentCommand \AutoRun { O{} } { \yoin_yoinshell_autorun:n { ##1 } }
200        \DeclareDocumentCommand \WriteMeta { O{} } { \yoin_yoinshell_writemeta:n { ##1 } }
201        \yoin_yoinshell_writemeta:n { }
202    }
```

```
203 }

204 \cs_new_protected:Nn \__yoin_yoinshell_end: {
205    \group_end:
206 }

207 \NewDocumentEnvironment { yoinshell } { O{} } {
208    \__yoin_yoinshell_begin:n { #1 }
209 } {
210    \__yoin_yoinshell_end:
211 }
```

6.1   RunForEach

\l__yoin_yoinshell_runforarticle_tag_tl
\l__yoin_yoinshell_runforeach_onlytag_tl

First, two `tls` that will store tags: One for the tag of the article, one that could be passed to `\RunForEach` that is initially set to `\q_no_value`.

```
212 \tl_new:N \l__yoin_yoinshell_runforarticle_tag_tl
213 \tl_new:N \l__yoin_yoinshell_runforeach_onlytag_tl
214 \tl_set:Nn \l__yoin_yoinshell_runforeach_onlytag_tl { \q_no_value }
```

yoin␣/␣runforeach

So far, the only key–val passable to `\RunForEach` is onlytag, which tests for the tag to be declared and passes it to `\l_@@_yoinshell_runforeach_only`

```
215 \keys_define:nn { yoin / runforeach } {
216    onlytag .code:n =
217       \__yoin_error_if_tag_undefined:n { #1 }
218       \tl_set:Nn \l__yoin_yoinshell_runforeach_onlytag_tl { #1 }
219       ,
220 }
```

__yoin_yoinshell_runforarticle_keyfromprop:nnN

This macro lets #3 to the value of property #2 of article #1. It makes it an empty definition if the property is unset.

```
221 \tl_new:N \l__yoin_yoinshell_runforarticle_tmpa_tl
222 \cs_new_protected:Nn \__yoin_yoinshell_runforarticle_keyfromprop:nnN {
223    \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinshell_runforarticle_tmpa_tl
224    \quark_if_no_value:NTF \l__yoin_yoinshell_runforarticle_tmpa_tl {
225       \def #3 {}
226    } {
227       \let #3 \l__yoin_yoinshell_runforarticle_tmpa_tl
228    }
229 }
```

__yoin_yoinshell_runforeach:nn — \RunForEach itself just sets the keys (in a group to make things local) and then calls \@@_yoinshell_runforarticle:nn on each article.

```
230 \cs_new_protected:Nn \yoin_yoinshell_runforeach:nn {
231     \group_begin:
232     \keys_set:nn { yoin / runforeach } { #1 }
233     \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinshell_runforarticle:nn { ##1 } { #2 } }
234     \group_end:
235 }
```

__yoin_yoinshell_runforarticle:nn — If the tag passed to onlytag of \RunForEach is identical to the tag of the article or if any of them is not set, we do what should be done, otherwise nothing is done (the tags do not match). We only extract the prop to publically available macros like \Article, \Jobname etc. (in a group to make this local), and then run the command in shell escape.

```
236 \cs_new_protected:Nn \__yoin_yoinshell_runforarticle:nn {
237     \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { tag } \l__yoin_yoinshell_runforarticle_tag_tl
238     \bool_if:nT {
239         \quark_if_no_value_p:N \l__yoin_yoinshell_runforarticle_tag_tl
240         ||
241         \quark_if_no_value_p:N \l__yoin_yoinshell_runforeach_onlytag_tl
242         ||
243         \tl_if_eq_p:NN \l__yoin_yoinshell_runforeach_onlytag_tl \l__yoin_yoinshell_runforarticle_tag_tl
244     }{
245         \group_begin:
246         \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { article } \Article
247         \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { jobname } \Jobname
248         \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { firstpage } \FirstPage
249         \__yoin_yoinshell_shellescape:n { #2 }
250         \group_end:
251     }
252 }
253 \cs_generate_variant:Nn \__yoin_yoinshell_runforarticle:nn { VV }
```

6.2   AutoRunForEach (undocumented)

```
254 \tl_new:N \l__yoin_yoinshell_autorunforeach_engine_tl
255 \tl_set:Nn \l__yoin_yoinshell_autorunforeach_engine_tl { \q_no_value }
256 \tl_new:N \l__yoin_yoinshell_autorunforeach_command_tl
257 \tl_new:N \l__yoin_yoinshell_autorunforeach_article_tl
258 \cs_new_protected:Nn \yoin_yoinshell_autorunforeach_new_engine:nn {
259     \cs_if_exist:cT { __yoin_yoinshell_autorunforeach_engine_preprocess_ #1 : } {
```

```
260        \msg_warning:nnn { yoin } { autorunforeach-duplicate-engine } { #1 }
261      }
262      \cs_new_protected:cn { __yoin_yoinshell_autorunforeach_engine_preprocess_ #1 : } { #2 }
263  }
264  \cs_new_protected:Nn \yoin_yoinshell_autorunforeach_new_variable:n {
265      \tl_new:c { l__yoin_yoinshell_autorunforeach_variable_ #1 _tl }
266      \keys_define:nn { yoin / autorunforeach } {
267          #1 .tl_set:c = { l__yoin_yoinshell_autorunforeach_variable_ #1 _tl } ,
268      }
269  }
270  \yoin_yoinshell_autorunforeach_new_engine:nn { latex }
271      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
272          { cd ~ "./\Article" ~ && ~ latex ~ -recorder ~ "./\Jobname" } }
273  \yoin_yoinshell_autorunforeach_new_engine:nn { dvips }
274      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
275          { cd ~ "./\Article" ~ && ~ dvips ~ "./\Jobname.dvi" } }
276  \yoin_yoinshell_autorunforeach_new_engine:nn { ps2pdf }
277      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
278          { cd ~ "./\Article" ~ && ~ ps2pdf ~ "./\Jobname.ps" } }
279  \yoin_yoinshell_autorunforeach_new_engine:nn { pdflatex }
280      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
281          { cd ~ "./\Article" ~ && ~ pdflatex ~ -recorder ~ "./\Jobname" } }
282  \yoin_yoinshell_autorunforeach_new_engine:nn { lualatex }
283      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
284          { cd ~ "./\Article" ~ && ~ lualatex ~ -recorder ~ "./\Jobname" } }
285  \yoin_yoinshell_autorunforeach_new_engine:nn { xelatex }
286      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
287          { cd ~ "./\Article" ~ && ~ xelatex ~ -recorder ~ "./\Jobname" } }
288  \yoin_yoinshell_autorunforeach_new_engine:nn { arara }
289      { \tl_set:Nn \l__yoin_yoinshell_autorunforeach_command_tl
290          { cd ~ "./\Article" ~ && ~ arara ~ "./\Jobname.tex" } }
291  \msg_new:nnn { yoin } { autorunforeach-noengine }
292      { Engine ~ unspecified ~ for ~ \token_to_str:N \AutoRunForEach . ~ I'm ~ trying  ~ 'pdflatex'. }
293  \msg_new:nnn { yoin } { autorunforeach-duplicate-engine }
294      { Engine ~ '#1' ~ defined ~ multiple ~ times. ~ Overwriting ~ the ~ first ~ definition. }
295  \msg_new:nnn { yoin } { autorunforeach-unknown-engine }
296      { Engine ~ '#1' ~ unknown. ~ I'm ~ trying  ~ 'pdflatex'. }
297  \keys_define:nn { yoin / autorunforeach } {
298      onlytag .code:n = \keys_set:nn { yoin / runforeach } { onlytag = { #1 } } ,
299      engine .tl_set:N = \l__yoin_yoinshell_autorunforeach_engine_tl ,
```

```
300 }
301 \cs_new_protected:Nn \yoin_yoinshell_autorunforeach:n {
302     \group_begin:
303     \keys_set:nn { yoin / autorunforeach } { #1 }
304     \quark_if_no_value:NT \l__yoin_yoinshell_autorunforeach_engine_tl {
305         \msg_error:nn { yoin } { autorunforeach-noengine }
306         \tl_set:Nn \l__yoin_yoinshell_autorunforeach_engine_tl { pdflatex }
307     }
308     \cs_if_exist:cF { __yoin_yoinshell_autorunforeach_engine_preprocess_ \l__yoin_yoinshell_autorunforeach_engine_tl : } {
309         \msg_error:nnx { yoin } { autorunforeach-unknown-engine } { \l__yoin_yoinshell_autorunforeach_engine_tl }
310         \tl_set:Nn \l__yoin_yoinshell_autorunforeach_engine_tl { pdflatex }
311     }
312     \seq_map_inline:Nn \g_yoin_yoinadd_seq {
313         \tl_set:Nn \l__yoin_yoinshell_autorunforeach_article_tl { ##1 }
314         \use:c { __yoin_yoinshell_autorunforeach_engine_preprocess_ \l__yoin_yoinshell_autorunforeach_engine_tl : }
315         \__yoin_yoinshell_runforarticle:VV
316             \l__yoin_yoinshell_autorunforeach_article_tl
317             \l__yoin_yoinshell_autorunforeach_command_tl
318     }
319     \group_end:
320 }
```

6.3 Run

__yoin_yoinshell_run:nn

```
321 \cs_new_protected:Nn \yoin_yoinshell_run:nn {
322     \group_begin:
323     \keys_set:nn { yoin / run } { #1 }
324     \let \Jobname \g_yoin_jobname_tl
325     \__yoin_yoinshell_shellescape:n { #2 }
326     \group_end:
327 }
```

6.4 AutoRun (undocumented)

```
328 \tl_new:N \l__yoin_yoinshell_autorun_engine_tl
329 \tl_set:Nn \l__yoin_yoinshell_autorun_engine_tl { \q_no_value }
330 \tl_new:N \l__yoin_yoinshell_autorun_command_tl
331 \cs_new_protected:Nn \yoin_yoinshell_autorun_new_engine:nn {
332     \cs_new_protected:cn { __yoin_yoinshell_autorun_engine_preprocess_ #1 : } { #2 }
```

```
333 }
334 \cs_new_protected:Nn \yoin_yoinshell_autorun_new_variable:n {
335    \tl_new:c { l__yoin_yoinshell_autorun_variable_ #1 _tl }
336    \keys_define:nn { yoin / autorun } {
337       #1 .tl_set:c = l__yoin_yoinshell_autorun_variable_ #1 _tl ,
338    }
339 }
340 \yoin_yoinshell_autorun_new_engine:nn { pdflatex } {
341    \tl_clear:N \l__yoin_yoinshell_autorun_command_tl
342    \tl_put_right:Nn \l__yoin_yoinshell_autorun_command_tl
343       { pdflatex ~ -recorder ~ -jobname ~ "\Jobname }
344    \tl_put_right:NV \l__yoin_yoinshell_autorun_command_tl
345       \l__yoin_yoinshell_autorun_variable_suffix_tl
346    \tl_put_right:Nn \l__yoin_yoinshell_autorun_command_tl
347       { " ~ "\noexpand\PassOptionsToPackage{subprocess, ~ jobname=\Jobname}{yoin}\noexpand\input{\Jobname}" }
348 }
349 \yoin_yoinshell_autorun_new_variable:n { suffix }
350 \msg_new:nnn { yoin } { autorun-noengine }
351    { Engine ~ unspecified ~ for ~ \token_to_str:N \AutoRun . ~ I'm ~ trying  ~ `pdflatex'. }
352 \msg_new:nnn { yoin } { autorun-unknown-engine }
353    { Engine ~ `#1' ~ unknown. ~ I'm ~ trying  ~ `pdflatex'. }
354 \keys_define:nn { yoin / autorun } {
355    engine .tl_set:N = \l__yoin_yoinshell_autorun_engine_tl ,
356 }
357 \cs_new_protected:Nn \yoin_yoinshell_autorun:n {
358    \group_begin:
359    \keys_set:nn { yoin / autorun } { #1 }
360    \quark_if_no_value:NT \l__yoin_yoinshell_autorun_engine_tl {
361       \msg_error:nn { yoin } { autorun-noengine }
362       \tl_set:Nn \l__yoin_yoinshell_autorun_engine_tl { pdflatex }
363    }
364    \cs_if_exist:cF { __yoin_yoinshell_autorun_engine_preprocess_ \l__yoin_yoinshell_autorun_engine_tl : } {
365       \msg_error:nnx { yoin } { autorun-unknown-engine } { \l__yoin_yoinshell_autorun_engine_tl }
366       \tl_set:Nn \l__yoin_yoinshell_autorun_engine_tl { pdflatex }
367    }
368    \use:c { __yoin_yoinshell_autorun_engine_preprocess_ \l__yoin_yoinshell_autorun_engine_tl : }
369    \let \Jobname \g_yoin_jobname_tl
370    \__yoin_yoinshell_shellescape:V \l__yoin_yoinshell_autorun_command_tl
371    \group_end:
372 }
```

6.5 WriteMeta

`\__yoin_yoinshell_writemeta:n`

```
373 \iow_new:N \g__yoin_yoinshell_iow
374 \cs_new_protected:Nn \yoin_yoinshell_writemeta:n {
375     \group_begin:
376     \yoin_yoinmeta:n { #1 }
377     \iow_open:Nn \g__yoin_yoinshell_iow { \g_yoin_jobname_tl .yoin }
378     \prop_map_inline:Nn \l__yoin_yoinmeta_prop {
379         \iow_now:Nn \g__yoin_yoinshell_iow { meta-##1 ~ = ~ { ##2 } , }
380     }
381     \iow_close:N \g__yoin_yoinshell_iow
382     \group_end:
383 }
```

7 macro yoinForEach

`\l__yoin_yoinforeach_article_tag_tl`
`\l__yoin_yoinforeach_onlytag_tl`

First, two `tls` that will store tags: One for the tag of the article, one that could be passed to `\yoinForEach` that is initially set to `\q_no_value`.

```
384 \tl_new:N \l__yoin_yoinforeach_article_tag_tl
385 \tl_new:N \l__yoin_yoinforeach_onlytag_tl
386 \tl_set:Nn \l__yoin_yoinforeach_onlytag_tl { \q_no_value }
```

`yoin␣/␣yoinforeach`

So far, the only key–val passable to `\yoinForEach` is onlytag, which tests for the tag to be declared and passes it to `\l_@@_yoinforeach_onlytag_tl`.

```
387 \keys_define:nn { yoin / yoinforeach } {
388     onlytag .code:n =
389         \__yoin_error_if_tag_undefined:n { #1 }
390         \tl_set:Nn \l__yoin_yoinforeach_onlytag_tl { #1 }
391         ,
392 }
```

`\__yoin_yoinforeach_article_keyfromprop:nnN`

This macro lets #3 to the value of property #2 of article #1. It makes it an empty definition if the property is unset.

```
393 \tl_new:N \l__yoin_yoinforeach_tmpa_tl
394 \cs_new_protected:Nn \__yoin_yoinforeach_article_keyfromprop:nnN {
395     \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinforeach_tmpa_tl
396     \quark_if_no_value:NTF \l__yoin_yoinforeach_article_tmpa_tl {
397         \def #3 {}
398     } {
```

```
399          \let #3 \l__yoin_yoinforeach_tmpa_tl
400       }
401 }
```

`\__yoin_yoinforeach_article_metaitem:nnN` This macro lets #3 to the value of property #2 of article #1. It makes it an empty definition if the property is unset.

```
402 \cs_new:Nn \__yoin_yoinforeach_article_metaitem:nn {
403     \yoin_yoinadd_prop_item:nn { #1 } { article-#2 }
404 }
```

`\__yoin_yoinforeach:nn` \yoinForEach itself just sets the keys (in a group to make things local) and then calls \@@_yoinforeach_article:nn on each article.

```
405 \cs_new_protected:Nn \yoin_yoinforeach:nn {
406     \group_begin:
407     \keys_set:nn { yoin / yoinforeach } { #1 }
408     \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinforeach_article:nn { ##1 } { #2 } }
409     \group_end:
410 }
```

`\__yoin_yoinshell_runforarticle:nn` If the tag passed to `onlytag` of \RunForEach is identical to the tag of the article or if any of them is not set, we do what should be done, otherwise nothing is done (the tags do not match). We only extract the prop to publically available macros like \Article, \Jobname etc. (in a group to make this local), and then run the command in shell escape.

```
411 \cs_new_protected:Nn \__yoin_yoinforeach_article:nn {
412     \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { tag } \l__yoin_yoinforeach_article_tag_tl
413     \bool_if:nT {
414         \quark_if_no_value_p:N \l__yoin_yoinforeach_article_tag_tl
415         ||
416         \quark_if_no_value_p:N \l__yoin_yoinforeach_onlytag_tl
417         ||
418         \tl_if_eq_p:NN \l__yoin_yoinforeach_onlytag_tl \l__yoin_yoinforeach_article_tag_tl
419     }
420     {
421         \group_begin:
422         \DeclareDocumentCommand \Meta { m } { \__yoin_yoinforeach_article_metaitem:nn { #1 } { ##1 } }
423         \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { article } \Article
424         \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { jobname } \Jobname
425         \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { firstpage } \FirstPage
426         #2
427         \group_end:
428     }
429 }
```

15

`\yoinForEach`   One optional key–val argument, one mandatory argument — the text to be typeset.

```
430 \NewDocumentCommand \yoinForEach { O{} +m } {
431     \yoin_yoinforeach:nn { #1 } { #2 }
432 }
```

8   Article setting stuff (undocumented)

Information to be stored in an auxiliary file.

```
433 \tl_new:N \l__yoin_article_tmpa_tl
434 \seq_new:N \l__yoin_article_tmpa_seq
435
436 \cs_new_protected:Nn \__yoin_article_write_keyval:nn {
437     \iow_now:Nn \g__yoin_article_dotyoin_iow { #1 ~ = ~ { #2 } , }
438 }
439 \cs_generate_variant:Nn \__yoin_article_write_keyval:nn { nx, nV }
440
441 \cs_new_protected:Nn \yoin_article_write_meta:nn {
442     \__yoin_article_write_keyval:nn { article-#1 } { #2 }
443 }
444
445 \cs_new_protected:Nn \yoin_article_write: {
446     \__yoin_article_write_keyval:nV { jobname } \g_yoin_jobname_tl
447     \__yoin_article_write_keyval:nx { totpages } { \ztotpages }
448     \__yoin_article_write_keyval:nV { currdir } \l_yoin_article_currdir_tl
449     \__yoin_article_write_keyval:nx { firstpage } { \int_use:N \l_yoin_article_firstpage_int }
450 }
451
452 \prop_new:N \l__yoin_article_read_prop
453
454 \cs_new_protected:Nn \yoin_article_read_put:nn {
455     \prop_put:Nnn \l__yoin_article_read_prop { #1 } { #2 }
456 }
457 \cs_generate_variant:Nn \yoin_article_read_put:nn { V }
458
459 \int_new:N \l_yoin_article_firstpage_int
460 \int_set:Nn \l_yoin_article_firstpage_int { 1 }
461
462 \keys_define:nn { yoin / toarticle } {
```

```
463    firstpage .code:n =
464        \int_set:Nn \l_yoin_article_firstpage_int { #1 }
465        \yoin_article_read_put:nn { firstpage } { #1 }
466        ,
467
468    parent .code:n =
469        \file_if_exist:nT { ../ #1 .yoin } {
470            \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ #1 .yoin }
471        }
472        \yoin_article_read_put:nn { parent } { #1 }
473        ,
474
475    unknown .code:n =
476        \yoin_article_read_put:Vn \l_keys_key_tl { #1 }
477        ,
478 }
479
480 \bool_new:N \g__yoin_article_read_bool
481
482 \cs_new_protected:Nn \yoin_article_read: {
483    \bool_if:NF \g__yoin_article_read_bool {
484        \file_if_exist:nT { ../ \l_yoin_article_currdir_tl .yoin1 } {
485            \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ \l_yoin_article_currdir_tl .yoin1 }
486        }
487    }
488    \bool_gset_true:N \g__yoin_article_read_bool
489 }
490
491 \cs_new:Nn \yoin_article_read_meta:n {
492    \prop_item:Nn \l__yoin_article_read_prop { meta-#1 }
493 }
494
495 \cs_new_protected:Nn \yoin_article_read_meta_gset_tl_default:Nnn {
496    \prop_get:NnNTF \l__yoin_article_read_prop { meta-#2 } \l__yoin_article_tmpa_tl {
497        \tl_gset_eq:NN #1 \l__yoin_article_tmpa_tl
498    } {
499        \tl_gset:Nn #1 { #3 }
500    }
501 }
502
```

```
503 \NewDocumentCommand \yoinArticleMeta { m } {
504     \yoin_article_read_meta:n { #1 }
505 }
506
507 \tl_new:N \l_yoin_article_currdir_tl
508 \cs_generate_variant:Nn \regex_extract_all:nnN { nV }
509 \cs_new_protected:Nn \yoin_article_getcurrdir:N {
510     \tl_set:Nx \l__yoin_article_tmpa_tl { \currfileabsdir / }
511     \regex_extract_all:nVN { /([^/][^/]+|[^/.]) } \l__yoin_article_tmpa_tl \l__yoin_article_tmpa_seq
512     \seq_get_right:NN \l__yoin_article_tmpa_seq #1
513 }
514
515 \iow_new:N \g__yoin_article_dotyoin_iow
516 \bool_if:NT \g_yoin_article_bool {
517     \yoin_article_getcurrdir:N \l_yoin_article_currdir_tl
518     \iow_open:Nn \g__yoin_article_dotyoin_iow { \l_yoin_article_currdir_tl .yoin }
519     \yoin_article_read:
520     \AtBeginDocument {
521         \setcounter { page } { \l_yoin_article_firstpage_int }
522         \yoin_article_write:
523     }
524 }
```

9   yoinProcess

Error message for an article having both `forceopenany` and `forceopenright` set.

```
525 \msg_new:nnn { yoin } { forceopenanyright }
526     { The ~ article ~ '#1' ~ has ~ both ~ 'forceopenany' ~ and ~ 'forceopenright' ~ keys ~ set. }
```

\yoin_yoinprocess: The cornerstone macro of the package, to some sense. It takes care of the page numbering of the articles, proper placement of stuff in twoside environment, etc.

```
527 \cs_new_protected:Nn \yoin_yoinprocess: {
```

Finish the current page if it's started.

```
528     \clearpage
```

Go to the right page number. This depends on two parameters, `cleardoublepage` and `setpagenumber`, the dependence is explained in each of the 4 cases.

```
529     \bool_if:NTF \l__yoin_yoinprocess_cleardoublepage_bool {
530         \bool_if:NTF \l__yoin_yoinprocess_setpagenumber_bool {
```

Case `cleardoublepage`, `setpagenumber`. In this case, an empty page is added as necessary to keep the parity of page numbers. For instance, if `setpagenumber=110` and last page number is 4, an empty page is added so that there are no two consecutive even pages. The check is on the parity of the sum of the two numbers. The macro `\__yoin_yoinprocess_clearonepage:` uses the code of LaTeX 2_ε's `\cleardoublepage` for creating the necessary empty page.

```
531          \int_if_odd:nT { \value { page } + \l__yoin_yoinprocess_setpagenumber_int } {
532              \__yoin_yoinprocess_clearonepage:
533          }
534          \setcounter { page } { \int_use:N \l__yoin_yoinprocess_setpagenumber_int }
535      } {
```

Case `cleardoublepage`, `nosetpagenumber`. We simply do a cleardoublepage. Note that `\__yoin_yoinprocess_cleardoublepage:` modifies the value of `\g_@@_page_int` in a useless way at this place, but we will override the value anyway.

```
536          \__yoin_yoinprocess_cleardoublepage:
537      }
538    } {
```

Case `nocleardoublepage`, `setpagenumber`. We simply set the page number.

```
539      \bool_if:NTF \l__yoin_yoinprocess_setpagenumber_bool {
540          \setcounter { page } { \int_use:N \l__yoin_yoinprocess_setpagenumber_int }
541      } {
```

Case `nocleardoublepage`, `nosetpagenumber`. No adjustment is needed in this case.

```
542          \prg_do_nothing:
543      }
544    }
```

Here, the loop through the articles starts. First, set the internal counter for the page number; this is necessary because if the output of the process is suppressed by the key `output=false`, we need to keep track of the page number manually.

```
545    \int_gset:Nn \g__yoin_page_int { \value { page } }
546    \seq_map_inline:Nn \g_yoin_yoinadd_seq {
```

Handing of even/odd/pages. First, issue an error if both `addarticle/forceopenany` and `addarticle/forceopenright` are set.

```
547      \bool_if:nT {
548        \int_compare_p:nNn {
549          \yoin_yoinadd_prop_item:nn { ##1 } { forceopenany }
550          + \yoin_yoinadd_prop_item:nn { ##1 } { forceopenright }
551        } = { 2 }
552      } {
553        \msg_error:nnn { yoin } { forceopenanyright } { ##1 }
554      }
```

Then, we call `cleardoublepage` (our internal variant) if: either `forceopenright` is true; or `openright` is true and `forceopenany` is false.

```
555        \bool_if:nT {
556            \int_compare_p:nNn { \yoin_yoinadd_prop_item:nn { ##1 } { forceopenright } } = { 1 }
557            || (
558                \l__yoin_yoinprocess_openright_bool
559                &&
560                \int_compare_p:nNn { \yoin_yoinadd_prop_item:nn { ##1 } { forceopenany } } = { 0 }
561            )
562        } {
563            \__yoin_yoinprocess_cleardoublepage:
564        }
```

If `output` is true, we use `\includepdf` to include the PDF of the article.

```
565        \bool_if:NT \l__yoin_yoinprocess_output_bool {
566            \includepdf [ pages = - ] { ##1 / \yoin_yoinadd_prop_item:nn { ##1 } { jobname } .pdf }
567        }
```

Into file `./<articlename>.yoin1` we save the data to be transferred to the article: the first page number (possibly 1 if `alwayspageone` key is set) and the name of the this document.

```
568        \iow_open:Nn \g__yoin_yoinprocess_iow { ##1 .yoin1 }
569        \bool_if:NTF \l__yoin_yoinprocess_alwayspageone_bool {
570            \iow_now:Nx \g__yoin_yoinprocess_iow { firstpage ~ = ~ { 1 } , }
571        } {
572            \iow_now:Nx \g__yoin_yoinprocess_iow { firstpage ~ = ~ { \int_use:N \g__yoin_page_int } , }
573        }
574        \iow_now:Nx \g__yoin_yoinprocess_iow { parent ~ = ~ { \g_yoin_jobname_tl } , }
575        \iow_close:N \g__yoin_yoinprocess_iow
```

Update our internal page counter.

```
576        \int_gadd:Nn \g__yoin_page_int { \yoin_yoinadd_prop_item:nn { ##1 } { totpages } }
577    }
578 }
```

\yoinProcess   Public wrapper around the LaTeX3 version.

```
579 \DeclareDocumentCommand \yoinProcess { } { \yoin_yoinprocess: }
```

\g__yoin_page_int   A private counter for tracking the page numbers, and an output stream for writing to `.yoin1` files.
\g__yoin_yoinprocess_iow

```
580 \int_new:N \g__yoin_page_int
581 \iow_new:N \g__yoin_yoinprocess_iow
```

__yoin_yoinprocess_cleardoublepage: If output is true, issue `\cleardoublepage`. Since this macro is always called after a page is finished (either after `\includepdf` or `\clearpage`), to correct the private page counter, we only need to round its value up to an odd number.

```
582 \cs_new_protected:Nn \__yoin_yoinprocess_cleardoublepage: {
583     \bool_if:NT \l__yoin_yoinprocess_output_bool { \cleardoublepage }
584     \int_if_even:nT { \g__yoin_page_int } { \int_gincr:N \g__yoin_page_int }
585 }
```

__yoin_yoinprocess_clearonepage: Clear exactly one page. Code borrowed from LaTeX 2$_\varepsilon$ kernel's `\cleardoublepage`.

```
586 \cs_new_protected:Nn \__yoin_yoinprocess_clearonepage: {
587     \bool_if:NT \l__yoin_yoinprocess_output_bool {
588         \hbox {}\newpage \if@twocolumn \hbox {}\newpage \fi
589     }
590     \int_gincr:N \g__yoin_page_int
591 }
```

\l__yoin_yoinprocess_cleardoublepage_bool Booleans and counters for values of the keys defined below.

\l__yoin_yoinprocess_output_bool
\l__yoin_yoinprocess_openright_bool
\l__yoin_yoinprocess_alwayspageone_bool
\l__yoin_yoinprocess_setpagenumber_bool
\l__yoin_yoinprocess_setpagenumber_int

```
592 \bool_new:N \l__yoin_yoinprocess_cleardoublepage_bool
593 \bool_new:N \l__yoin_yoinprocess_output_bool
594 \bool_new:N \l__yoin_yoinprocess_openright_bool
595 \bool_new:N \l__yoin_yoinprocess_alwayspageone_bool
596 \bool_new:N \l__yoin_yoinprocess_setpagenumber_bool
597 \int_new:N \l__yoin_yoinprocess_setpagenumber_int
```

yoin␣/␣general Keys for `yoinprocess`: several boolean keys (including `openany` as the negation of `openright`), and `setpagenumber`, taking as a value either a number or `false` (if a number is input, it is stored in a counter with the appropriate boolean set `true`). We actually define these keys as general keys and they are set up either when loading the package of using `\yoinSetup`, they are defined here because they basically only interact with `\yoinProcess`.

```
598 \keys_define:nn { yoin / general } {

599     cleardoublepage .bool_set:N = \l__yoin_yoinprocess_cleardoublepage_bool ,
600     cleardoublepage .initial:n = { false },

601     output .bool_set:N = \l__yoin_yoinprocess_output_bool ,
602     output .initial:n = { true },

603     openright .bool_set:N = \l__yoin_yoinprocess_openright_bool ,
604     openany .bool_set_inverse:N = \l__yoin_yoinprocess_openright_bool ,
605     openright .initial:n = { false },

606     alwayspageone .bool_set:N = \l__yoin_yoinprocess_alwayspageone_bool ,
607     alwayspageone .initial:n = { false },
```

```
608    setpagenumber .code:n =
609        \str_if_eq:nnTF { #1 } { false } {
610            \bool_set_false:N \l__yoin_yoinprocess_setpagenumber_bool
611        } {
612            \bool_set_true:N \l__yoin_yoinprocess_setpagenumber_bool
613            \int_set:Nn \l__yoin_yoinprocess_setpagenumber_int { #1 }
614        }
615        ,
616    setpagenumber .initial:n = { false },

617 }
```

10   Process keys

\ProcessKeysPackageOptions   Process key options given to the package. We *do not want to process any options given to the class.* Whence \ProcessKeysPackageOptions and not \ProcessKeysOptions.

```
618 \ProcessKeysPackageOptions { yoin / general }
```

11   Experimental

\bla

```
619 \cs_new:Nn \yoin_blabla: {
620     Blabla
621 }
622

623 ⟨/package⟩
```