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 {
```

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

msg: boolean-values-only

Message for a non-boolean passed to a bool key.

```
\msg_new:nnn { yoin } { boolean-values-only }
    { 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
\g_yoin_article_bool
\g_yoin_dryrun_bool
\g_yoin_onlyflags_bool
\g_yoin_onlytags_bool

Booleans:

```
\bool_new:N \g_yoin_subprocess_bool
\bool_new:N \g_yoin_article_bool
\bool_new:N \g_yoin_dryrun_bool
\bool_new:N \g_yoin_onlyflags_bool
\bool_new:N \g_yoin_onlytags_bool
```

\g_yoin_flags_seq
\g_yoin_tags_seq
\g_yoin_onlyflags_seq
\g_yoin_onlytags_seq

Sequences for flags, tags and their filtering:

```
\seq_new:N \g_yoin_flags_seq
\seq_new:N \g_yoin_tags_seq
\seq_new:N \g_yoin_onlyflags_seq
\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:

```
\tl_new:N \g_yoin_jobname_tl
\tl_gset_eq:NN \g_yoin_jobname_tl \c_job_name_tl
```

msg: unknown-flag
msg: unknown-tag

Two messages, for unknown flags and unknown tags.

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

\msg_new:nnnn { yoin } { unknown-tag }
    { The ~ tag ~ `#1' ~ is ~ unknown ~ to ~ `yoin'. }
    { 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.

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

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

__yoin_error_if_tag_undefined:n
__yoin_error_if_flag_undefined:n

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

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

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

yoin / general

The keys themselves:

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

Booleans:

```
   dryrun .bool_gset:N = \g_yoin_dryrun_bool,
   dryrun .initial:n = { false },

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

   subprocess .bool_gset:N = \g_yoin_subprocess_bool,
   subprocess .initial:n = { false },
```

Keys whose `clist` values are appended to a `seq`:

```
   defineflags .code:n = \yoin_seq_gappend_clist:Nn \g_yoin_flags_seq { #1 },

   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.)

```
   onlyflags .code:n =
      \seq_gset_from_clist:Nn \g_yoin_onlyflags_seq { #1 }
      \bool_gset_true:N \g_yoin_onlyflags_bool
      ,
```

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

A key whose value is stored in a token list.

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

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

}
```

`\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`.

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

`\yoin_setup:n` `\yoinSetup` Allow keys to be set later. We define both a LaTeX3 interface and an xparse UI wrapper.

```
\cs_new_protected:Nn \yoin_setup:n {
   \keys_set:nn { yoin / general } { #1 }
}

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

4 yoinMeta macro — adding issue's metadata

`\yoinMeta` `\yoin_yoinmeta:n`

```
\prop_new:N \l__yoin_yoinmeta_prop
\cs_new_protected:Nn \__yoin_yoinmeta_storekey:nn {
   \prop_put:Nnn \l__yoin_yoinmeta_prop { #1 } { #2 }
}
\cs_new_protected:Nn \__yoin_yoinmeta_storekey:n {
   \prop_put:Nnn \l__yoin_yoinmeta_prop { #1 } { }
}
\cs_new_protected:Nn \yoin_yoinmeta:n {
   \keyval_parse:NNn \__yoin_yoinmeta_storekey:n \__yoin_yoinmeta_storekey:nn { #1 }
```

```
}
\NewDocumentCommand \yoinMeta { R[]{} } {
   \yoin_yoinmeta:n { #1 }
}
```

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.

```
\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.

```
\cs_new:Nn \yoin_yoinadd_prop:n {
   g__yoin_article_#1_prop
}
\cs_generate_variant:Nn \yoin_yoinadd_prop:n { V }

\cs_new:Nn \yoin_yoinadd_prop_item:nn {
   \prop_item:cn { \yoin_yoinadd_prop:n { #1 } } { #2 }
}
\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`.

```
\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.

```
\cs_new_protected:Nn \__yoin_yoinadd_storekey:nn {
   \prop_gput:cnn { \yoin_yoinadd_prop:V \l__yoin_yoinadd_currentarticle_tl } { #1 } { #2 }
}
\cs_new_protected:Nn \__yoin_yoinadd_storekey:n {
   \prop_gput:cnn { \yoin_yoinadd_prop:V \l__yoin_yoinadd_currentarticle_tl } { #1 } { }
}
```

\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).

```
\cs_new_protected:Nn \yoin_yoinadd:nn {
   \tl_set:Nn \l__yoin_yoinadd_currentarticle_tl { #1 }
   \seq_if_in:NnTF \g_yoin_yoinadd_seq { #1 } {
      \msg_error:nnn { yoin } { yoinadd-duplicatearticle } { #1 }
   } {
      \seq_gput_right:Nn \g_yoin_yoinadd_seq { #1 }
      \prop_new:c { \yoin_yoinadd_prop:n { #1 } }
      \clist_map_inline:nn { forceopenany, forceopenright, ignore } {
         \__yoin_yoinadd_storekey:nn { ##1 } { 0 }
      }
      \__yoin_yoinadd_storekey:nn { article } { #1 }
      \keys_set:nn { yoin / yoinadd } { #2 }
      \file_if_exist:nTF { #1 / #1 .yoin } {
         \yoin_keyval_parse_from_file:NNn
            \__yoin_yoinadd_storekey:n
            \__yoin_yoinadd_storekey:nn
            { #1 / #1 .yoin }
      } {
         \msg_error:nnn { yoin } { yoinadd-dotyoinmissing } { #1 }
      }
   }
}

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

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.

```
\msg_new:nnn { yoin } { yoinadd-duplicatearticle }
   { The ~ article ~ "#1" ~ has ~ been ~ already ~ processed ~ by ~ \token_to_str:N \yoinAdd ~.}
\msg_new:nnn { yoin } { yoinadd-dotyoinmissing }
   { 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.

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

For boolean keys, we create a manual boolean parser.

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

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

```
\keys_define:nn { yoin / yoinadd } {
   tag .code:n =
      \__yoin_error_if_tag_undefined:n { #1 }
      \__yoin_yoinadd_storekey:nn { tag } { #1 }
      ,
}
```

6 Environment yoinshell

`\l_yoin_yoinshell_ignore_bool` A boolean for storing the `ignore` key's value.

`yoin / yoinshell` Key–value interface to `yoinshell`.

```
\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).

```
   flag .code:n =
      \__yoin_error_if_flag_undefined:n { #1 }
      \bool_if:NT \g_yoin_onlyflags_bool {
         \seq_if_in:NnF \g_yoin_onlyflags_seq { #1 } {
            \keys_set:nn { yoin / yoinshell } {
               ignore = true
            }
         }
```

```
      }
      ,
```

The `ignore` key sets a boolean

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

shellesc.sty
\ShellEscape
__yoin_yoinshell_shellescape:n

A reasonable shell escape that should work in both `pdflatex` and `lualatex` in TeX Live 2016.

```
\file_if_exist:nTF { shellesc.sty } {
   \RequirePackage { shellesc }
} {
   \def \ShellEscape #1 { \immediate \write 18 { #1 } }
}
\cs_new_protected:Nn \__yoin_yoinshell_shellescape:n {
   \ShellEscape { #1 }
}
```

__yoin_yoinshell_begin:n
__yoin_yoinshell_end:
{yoinshell}

Environment `yoinshell` (one key–value argument). We perform some local definitions that should stay local, so we put everything 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 package, these macros are declared to do nothing, otherwise they are simply wrappers to the LaTeX3 counterparts.

```
\cs_new_protected:Nn \__yoin_yoinshell_begin:n {
   \group_begin:
   \keys_set:nn { yoin / yoinshell } { #1 }
   \bool_if:NT \g_yoin_subprocess_bool {
      \bool_set_true:N \l_yoin_yoinshell_ignore_bool
   }
   \bool_if:NTF \l_yoin_yoinshell_ignore_bool {
      \DeclareDocumentCommand \RunForEach { O{} m } { }
      \DeclareDocumentCommand \Run { O{} m } { }
      \DeclareDocumentCommand \WriteMeta { O{} } { }
   } {
      \DeclareDocumentCommand \RunForEach { O{} m } { \yoin_yoinshell_runforeach:nn { ##1 } { ##2 } }
      \DeclareDocumentCommand \Run { O{} m } { \yoin_yoinshell_run:nn { ##1 } { ##2 } }
      \DeclareDocumentCommand \WriteMeta { O{} } { \yoin_yoinshell_writemeta:n { ##1 } }
      \yoin_yoinshell_writemeta:n { }
   }
}
```

```
\cs_new_protected:Nn \__yoin_yoinshell_end: {
   \group_end:
}

\NewDocumentEnvironment { yoinshell } { O{} } {
   \__yoin_yoinshell_begin:n { #1 }
} {
   \__yoin_yoinshell_end:
}
```

6.1 RunForEach

\l__yoin_yoinshell_runforarticle_tag_tl
\l__yoin_yoinshell_runforeach_onlytag_tl

First, two `tl`s 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`.

```
\tl_new:N \l__yoin_yoinshell_runforarticle_tag_tl
\tl_new:N \l__yoin_yoinshell_runforeach_onlytag_tl
\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`

```
\keys_define:nn { yoin / runforeach } {
   onlytag .code:n =
      \__yoin_error_if_tag_undefined:n { #1 }
      \tl_set:Nn \l__yoin_yoinshell_runforeach_onlytag_tl { #1 }
      ,
}
```

__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.

```
\tl_new:N \l__yoin_yoinshell_runforarticle_tmpa_tl
\cs_new_protected:Nn \__yoin_yoinshell_runforarticle_keyfromprop:nnN {
   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinshell_runforarticle_tmpa_tl
   \quark_if_no_value:NTF \l__yoin_yoinshell_runforarticle_tmpa_tl {
      \def #3 {}
   } {
      \let #3 \l__yoin_yoinshell_runforarticle_tmpa_tl
   }
}
```

__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.

```
\cs_new_protected:Nn \yoin_yoinshell_runforeach:nn {
   \group_begin:
   \keys_set:nn { yoin / runforeach } { #1 }
   \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinshell_runforarticle:nn { ##1 } { #2 } }
   \group_end:
}
```

__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.

```
\cs_new_protected:Nn \__yoin_yoinshell_runforarticle:nn {
   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { tag } \l__yoin_yoinshell_runforarticle_tag_tl
   \bool_if:nT {
      \quark_if_no_value_p:N \l__yoin_yoinshell_runforarticle_tag_tl
      ||
      \quark_if_no_value_p:N \l__yoin_yoinshell_runforeach_onlytag_tl
      ||
      \tl_if_eq_p:NN \l__yoin_yoinshell_runforeach_onlytag_tl \l__yoin_yoinshell_runforarticle_tag_tl
   }{
      \group_begin:
      \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { article } \Article
      \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { jobname } \Jobname
      \__yoin_yoinshell_runforarticle_keyfromprop:nnN { #1 } { firstpage } \FirstPage
      \__yoin_yoinshell_shellescape:n { #2 }
      \group_end:
   }
}
```

6.2 Run

__yoin_yoinshell_run:nn

```
\cs_new_protected:Nn \yoin_yoinshell_run:nn {
   \group_begin:
   \keys_set:nn { yoin / run } { #1 }
   \let \Jobname \c_job_name_tl
   \__yoin_yoinshell_shellescape:n { #2 }
   \group_end:
```

```
}
```

6.3 WriteMeta

__yoin_yoinshell_writemeta:n

```
\iow_new:N \g__yoin_yoinshell_iow
\cs_new_protected:Nn \yoin_yoinshell_writemeta:n {
   \group_begin:
   \yoin_yoinmeta:n { #1 }
   \iow_open:Nn \g__yoin_yoinshell_iow { \g_yoin_jobname_tl .yoin }
   \prop_map_inline:Nn \l__yoin_yoinmeta_prop {
      \iow_now:Nn \g__yoin_yoinshell_iow { meta-##1 ~ = ~ ##2, }
   }
   \iow_close:N \g__yoin_yoinshell_iow
   \group_end:
}
```

7 macro yoinForEach

\l__yoin_yoinforeach_article_tag_tl
\l__yoin_yoinforeach_onlytag_tl

First, two `tl`s 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`.

```
\tl_new:N \l__yoin_yoinforeach_article_tag_tl
\tl_new:N \l__yoin_yoinforeach_onlytag_tl
\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`.

```
\keys_define:nn { yoin / yoinforeach } {
   onlytag .code:n =
      \__yoin_error_if_tag_undefined:n { #1 }
      \tl_set:Nn \l__yoin_yoinforeach_onlytag_tl { #1 }
      ,
}
```

__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.

```
\tl_new:N \l__yoin_yoinforeach_tmpa_tl
\cs_new_protected:Nn \__yoin_yoinforeach_article_keyfromprop:nnN {
   \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { #2 } \l__yoin_yoinforeach_tmpa_tl
```

```
    \quark_if_no_value:NTF \l__yoin_yoinforeach_article_tmpa_tl {
       \def #3 {}
    } {
       \let #3 \l__yoin_yoinforeach_tmpa_tl
    }
}
```

__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.

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

__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.

```
\cs_new_protected:Nn \yoin_yoinforeach:nn {
    \group_begin:
    \keys_set:nn { yoin / yoinforeach } { #1 }
    \seq_map_inline:Nn \g_yoin_yoinadd_seq { \__yoin_yoinforeach_article:nn { ##1 } { #2 } }
    \group_end:
}
```

__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.

```
\cs_new_protected:Nn \__yoin_yoinforeach_article:nn {
    \prop_get:cnN { \yoin_yoinadd_prop:n { #1 } } { tag } \l__yoin_yoinforeach_article_tag_tl
    \bool_if:nT {
       \quark_if_no_value_p:N \l__yoin_yoinforeach_article_tag_tl
       ||
       \quark_if_no_value_p:N \l__yoin_yoinforeach_onlytag_tl
       ||
       \tl_if_eq_p:NN \l__yoin_yoinforeach_onlytag_tl \l__yoin_yoinforeach_article_tag_tl
    }
    {
       \group_begin:
       \DeclareDocumentCommand \Meta { m } { \__yoin_yoinforeach_article_metaitem:nn { #1 } { ##1 } }
       \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { article } \Article
       \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { jobname } \Jobname
       \__yoin_yoinforeach_article_keyfromprop:nnN { #1 } { firstpage } \FirstPage
       #2
```

```
        \group_end:
    }
}
```

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

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

8 Article setting stuff (undocumented)

Information to be stored in an auxiliary file.

```
\cs_new_protected:Nn \__yoin_article_write_keyval:nn {
    \iow_now:Nn \g__yoin_article_dotyoin_iow { #1 ~ = ~ #2 , }
}
\cs_generate_variant:Nn \__yoin_article_write_keyval:nn { nx, nV }

\cs_new_protected:Nn \yoin_article_write_meta:nn {
    \__yoin_article_write_keyval:nn { article-#1 } { #2 }
}

\cs_new_protected:Nn \yoin_article_writekeys: {
    \__yoin_article_write_keyval:nV { jobname } \c_job_name_tl
    \__yoin_article_write_keyval:nx { totpages } { \ztotpages }
    \__yoin_article_write_keyval:nV { currdir } \l_yoin_article_currdir_tl
    \__yoin_article_write_keyval:nx { firstpage } { \int_use:N \l_yoin_article_firstpage_int }
}

\prop_new:N \l__yoin_article_readkeys_prop

\cs_new_protected:Nn \yoin_article_set_readkey:nn {
    \prop_put:Nnn \l__yoin_article_readkeys_prop { #1 } { #2 }
}
\cs_generate_variant:Nn \yoin_article_set_readkey:nn { Vn }

\int_new:N \l_yoin_article_firstpage_int
\int_set:Nn \l_yoin_article_firstpage_int { 1 }

\keys_define:nn { yoin / toarticle } {
```

```
    firstpage .code:n =
      \int_set:Nn \l_yoin_article_firstpage_int { #1 }
      \yoin_article_set_readkey:nn { firstpage } { #1 }
      ,

    parent .code:n =
      \file_if_exist:nT { ../ #1 .yoin } {
        \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ #1 .yoin }
      }
      \yoin_article_set_readkey:nn { parent } { #1 }
      ,

    unknown .code:n =
      \yoin_article_set_readkey:Vn \l_keys_key_tl { #1 }
      ,
}

\bool_new:N \g__yoin_article_readkeys_bool
\bool_gset_true:N \g__yoin_article_readkeys_bool

\cs_new_protected:Nn \yoin_article_readkeys: {
    \bool_if:NT \g__yoin_article_readkeys_bool {
      \file_if_exist:nT { ../ \l_yoin_article_currdir_tl .yoin1 } {
        \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ \l_yoin_article_currdir_tl .yoin1 }
      }
    }
    \bool_gset_false:N \g__yoin_article_readkeys_bool
}

\cs_new:Nn \yoin_article_meta:n {
    \prop_item:Nn \l__yoin_article_readkeys_prop { meta-#1 }
}

\cs_new:Nn \yoin_article_meta_gset_tl_default:Nnn {
    \prop_get:NnNF \l__yoin_article_readkeys_prop { meta-#2 } #1 {
      \tl_set:Nn #1 { #3 }
    }
}

\NewDocumentCommand \yoinArticleMeta { m } {
```

```
    \yoin_article_meta:n { #1 }
}

\tl_new:N \l__yoin_article_tmpa_tl
\seq_new:N \l__yoin_article_tmpa_seq
\tl_new:N \l_yoin_article_currdir_tl
\cs_generate_variant:Nn \regex_extract_once:nnN { nV }
\cs_new_protected:Nn \yoin_article_getcurrdir:N {
    \tl_set:Nx \l__yoin_article_tmpa_tl { \currfileabsdir }
    \regex_extract_once:nVN { /([^/]+)/\Z } \l__yoin_article_tmpa_tl \l__yoin_article_tmpa_seq
    \seq_get_right:NN \l__yoin_article_tmpa_seq #1
}

\iow_new:N \g__yoin_article_dotyoin_iow
\bool_if:NT \g_yoin_article_bool {
    \yoin_article_getcurrdir:N \l_yoin_article_currdir_tl
    \iow_open:Nn \g__yoin_article_dotyoin_iow { \l_yoin_article_currdir_tl .yoin }
    \yoin_article_readkeys:
    \AtBeginDocument {
        \setcounter { page } { \l_yoin_article_firstpage_int }
        \yoin_article_writekeys:
    }
}
```

9 yoinProcess (undocumented)

msg: forceopenanyright

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

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

\yoin_yoinprocess:n

The key 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.

```
\cs_new_protected:Nn \yoin_yoinprocess:n {
```

Set the appropriate keys (this may be changed later and moved to `yoin/general` keys.

```
    \keys_set:nn { yoin / yoinprocess } { #1 }
```

Finish the current page if it's started.

```
    \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.

```
    \bool_if:NTF \l__yoin_yoinprocess_cleardoublepage_bool {
      \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$_\varepsilon$'s `\cleardoublepage` for creating the necessary empty page.

```
        \int_if_odd:nT { \value { page } + \l__yoin_yoinprocess_setpagenumber_int } {
          \__yoin_yoinprocess_clearonepage:
        }
        \setcounter { page } { \int_use:N \l__yoin_yoinprocess_setpagenumber_int }
      } {
```

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.

```
        \__yoin_yoinprocess_cleardoublepage:
      }
    } {
```

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

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

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

```
        \prg_do_nothing:
      }
    }
```

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.

```
    \int_gset:Nn \g__yoin_page_int { \value { page } }
    \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.

```
      \bool_if:nT {
        \int_compare_p:nNn {
          \yoin_yoinadd_prop_item:nn { ##1 } { forceopenany }
          + \yoin_yoinadd_prop_item:nn { ##1 } { forceopenright }
        } = { 2 }
      } {
```

```
435         \msg_error:nnn { yoin } { forceopenanyright } { ##1 }
436       }
```

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

```
437       \bool_if:nT {
438         \int_compare_p:nNn { \yoin_yoinadd_prop_item:nn { ##1 } { forceopenright } } = { 1 }
439         || (
440           \l__yoin_yoinprocess_openright_bool
441           &&
442           \int_compare_p:nNn { \yoin_yoinadd_prop_item:nn { ##1 } { forceopenany } } = { 0 }
443         )
444       } {
445         \__yoin_yoinprocess_cleardoublepage:
446       }
```

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

```
447       \bool_if:NT \l__yoin_yoinprocess_output_bool {
448         \includepdf [ pages = - ] { ##1 / \yoin_yoinadd_prop_item:nn { ##1 } { jobname } .pdf }
449       }
```

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.

```
450       \iow_open:Nn \g__yoin_yoinprocess_iow { ##1 .yoin1 }
451       \bool_if:NTF \l__yoin_yoinprocess_alwayspageone_bool {
452         \iow_now:Nx \g__yoin_yoinprocess_iow { firstpage ~ = ~ 1 , }
453       } {
454         \iow_now:Nx \g__yoin_yoinprocess_iow { firstpage ~ = ~ \int_use:N \g__yoin_page_int , }
455       }
456       \iow_now:Nx \g__yoin_yoinprocess_iow { parent ~ = ~ \jobname , }
457       \iow_close:N \g__yoin_yoinprocess_iow
```

Update our internal page counter.

```
458       \int_gadd:Nn \g__yoin_page_int { \yoin_yoinadd_prop_item:nn { ##1 } { totpages } }
459     }
460 }
```

`\yoinProcess` Public wrapper around the LaTeX3 version.

```
461 \DeclareDocumentCommand \yoinProcess { O{} } { \yoin_yoinprocess:n { #1 } }
```

\g__yoin_page_int
\g__yoin_yoinprocess_iow

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

```
\int_new:N \g__yoin_page_int
\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.

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

__yoin_yoinprocess_clearonepage:

Clear exactly one page. Code borrowed from LaTeX 2_ε kernel's `\cleardoublepage`.

```
\cs_new_protected:Nn \__yoin_yoinprocess_clearonepage: {
   \bool_if:NT \l__yoin_yoinprocess_output_bool {
      \hbox {}\newpage \if@twocolumn \hbox {}\newpage \fi
   }
   \int_gincr:N \g__yoin_page_int
}
```

\l__yoin_yoinprocess_cleardoublepage_bool
\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

Booleans and counters for values of the keys defined below.

```
\bool_new:N \l__yoin_yoinprocess_cleardoublepage_bool
\bool_new:N \l__yoin_yoinprocess_output_bool
\bool_new:N \l__yoin_yoinprocess_openright_bool
\bool_new:N \l__yoin_yoinprocess_alwayspageone_bool
\bool_new:N \l__yoin_yoinprocess_setpagenumber_bool
\int_new:N \l__yoin_yoinprocess_setpagenumber_int
```

yoin / yoinprocess

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`).

```
\keys_define:nn { yoin / yoinprocess } {

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

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

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

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

    setpagenumber .code:n =
       \str_if_eq:nnTF { #1 } { false } {
          \bool_set_false:N \l__yoin_yoinprocess_setpagenumber_bool
       } {
          \bool_set_true:N \l__yoin_yoinprocess_setpagenumber_bool
          \int_set:Nn \l__yoin_yoinprocess_setpagenumber_int { #1 }
       }
       ,
    setpagenumber .initial:n = { false },

}
```

10 Experimental

\bla

```
\cs_new:Nn \yoin_blabla: {
   Blabla
}

```

⟨/package⟩