1 Package header

```
⟨*package⟩
⟨@@=yoin⟩
```

Necessary packages: First, LaTeX3 stuff.

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

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

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

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

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

For including PDF files.

```
\RequirePackage{pdfpages}
```

Package header.

```
\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`.

```
\seq_new:N \l__yoin_seq_tmpa_seq
\cs_new_protected:Nn \yoin_seq_gappend_clist:Nn {
   \seq_set_from_clist:Nn \l__yoin_seq_tmpa_seq { #2 }
   \seq_gconcat:NNN #1 #1 \l__yoin_seq_tmpa_seq
}
```

`\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).

```
\tl_new:N \l__yoin_keys_tmpa_tl
\cs_new_protected:Nn \yoin_keys_set_from_file:nn {
   \tl_set_from_file:Nnn \l__yoin_keys_tmpa_tl { } { #2 }
   \keys_set:nV { #1 } \l__yoin_keys_tmpa_tl
}
\cs_generate_variant:Nn \keys_set:nn { nV }
```

\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).

```
\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
}
\cs_generate_variant:Nn \keyval_parse:NNn { NNV }
```

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
```

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
      ,
}
```

\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 \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 } }
      \__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 { forceopenany, forceopenright, ignore } {
   \keys_define:nn { yoin / yoinadd } {
      #1 .code:n = \__yoin_yoinadd_storekey:nn { #1 } { ##1 },
   }
}
```

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 }
      ,
}
```

5 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, 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:NTF \l_yoin_yoinshell_ignore_bool {
        \DeclareDocumentCommand \RunForEach { O{} m } { }
        \DeclareDocumentCommand \Run { O{} m } { }
    } {
        \DeclareDocumentCommand \RunForEach { O{} m } { \yoin_yoinshell_runforeach:nn { ##1 } { ##2 } }
        \DeclareDocumentCommand \Run { O{} m } { \yoin_yoinshell_run:nn { ##1 } { ##2 } }
    }
}

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

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

The yoinshell command \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.

```
\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_onlytag_tl.

```
\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_shellescape:n { #2 }
      \group_end:
   }
}
```

6 Article setting stuff (undocumented)

Information to be stored in an auxiliary file.

```
\cs_new_protected:Nn \__yoin_article_write:n {
   \immediate \write \@auxout { \token_to_str:N \@writefile { yoin } { #1 } }
}

\cs_new_protected:Nn \__yoin_article_write_keyval:nn {
   \__yoin_article_write:n { #1 ~ = ~ #2 , }
}
\cs_generate_variant:Nn \__yoin_article_write_keyval:nn { nx }

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

\cs_new_protected:Nn \yoin_article_writekeys: {
   \__yoin_article_write_keyval:nx { jobname } { \jobname }
   \__yoin_article_write_keyval:nx { totpages } { \ztotpages }
   \__yoin_article_write_keyval:nx { 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 }
}

\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:nn { \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 .yoin } {
         \yoin_keys_set_from_file:nn { yoin / toarticle } { ../ \l_yoin_article_currdir_tl .yoin }
      }
```

```
  }
  \bool_gset_false:N \g__yoin_article_readkeys_bool
}

\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_new_protected:Nn \yoin_article_getcurrdir:N {
  \tl_set:Nx \l__yoin_article_tmpa_tl { \currfileabsdir }
  \cs_generate_variant:Nn \regex_extract_once:nnNF { nV }
  \regex_extract_once:nVNF { /([^/]+)/\Z } \l__yoin_article_tmpa_tl \l__yoin_article_tmpa_seq { \error }
  \seq_get_right:NN \l__yoin_article_tmpa_seq #1
}

\AtBeginDocument{ \yoin_atbegindocument: }

\cs_new_protected:Nn \yoin_atbegindocument: {
  \expandafter \newwrite \csname tf@yoin\endcsname
  \bool_if:NTF \g_yoin_article_bool {
    \yoin_article_getcurrdir:N \l_yoin_article_currdir_tl
    \immediate \openout \csname tf@yoin\endcsname \l_yoin_article_currdir_tl .yoin\relax
    \yoin_article_readkeys:
    \setcounter { page } { \l_yoin_article_firstpage_int }
    \yoin_article_writekeys:
  } {
    \immediate \openout \csname tf@yoin\endcsname \jobname .yoin\relax
  }
}
```

7 yoinProcess (undocumented)

```
\int_new:N \g_yoin_page_int
\iow_new:N \g__yoin_yoinprocess_stream
\cs_new_protected:Nn \yoin_yoinprocess:n {
  \keys_set:nn { yoin / yoinprocess } { #1 }
  \clearpage
  \message{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA}
  \bool_if:NTF \l__yoin_yoinprocess_cleardoublepage_bool {
    \bool_if:NTF \l__yoin_yoinprocess_setpagenumber_bool {
          %^^A CDP + SPN
          \message{(((CDP+SPN)))}
          \int_if_odd:nT { \value { page } + \l__yoin_yoinprocess_setpagenumber_int } {
             \bool_if:NT \l__yoin_yoinprocess_output_bool {
                \hbox {}\newpage \if@twocolumn \hbox {}\newpage \fi
             }
          }
          \setcounter { page } { \int_use:N \l__yoin_yoinprocess_setpagenumber_int }
       } {
          %^^A CDP
          \message{(((CDP)))}
          \__yoin_yoinprocess_cleardoublepage:
       }
    } {
       \bool_if:NTF \l__yoin_yoinprocess_setpagenumber_bool {
          %^^A SPN
          \message{(((SPN)))}
          \setcounter { page } { \int_use:N \l__yoin_yoinprocess_setpagenumber_int }
       } {
          %^^A neither
          \message{((()))}
       }
    }
    \int_gset:Nn \g_yoin_page_int { \value { page } }
    \seq_map_inline:Nn \g_yoin_yoinadd_seq {
       \bool_if:NT \l__yoin_yoinprocess_openright_bool {
          \__yoin_yoinprocess_cleardoublepage:
       }
       \bool_if:NT \l__yoin_yoinprocess_output_bool {
          \includepdf [ pages = - ] { ##1 / \yoin_yoinadd_prop_item:nn { ##1 } { jobname } .pdf }
       }
       \iow_open:Nn \g__yoin_yoinprocess_stream { ##1 .yoin }
       \iow_now:Nx \g__yoin_yoinprocess_stream { firstpage ~ = ~ \int_use:N \g_yoin_page_int , }
       \iow_now:Nx \g__yoin_yoinprocess_stream { parent ~ = ~ \jobname , }
       \iow_close:N \g__yoin_yoinprocess_stream
       \int_gadd:Nn \g_yoin_page_int { \yoin_yoinadd_prop_item:nn { ##1 } { totpages } }
    }
 }
 \DeclareDocumentCommand \yoinProcess { O{} } { \yoin_yoinprocess:n { #1 } }

```

```
\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 }
}

\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_setpagenumber_bool
\int_new:N \l__yoin_yoinprocess_setpagenumber_int
\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 },

   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 },

}

```

8 Experimental

\bla

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

```

⟨/package⟩
```