The morewrites package: Always room for a new \write*

Bruno Le Floch

July 10, 2012

Contents

*This file has version number v0.2, last revised 2012-07-10.

1 morewrites documentation

This LaTeX package is meant to be a solution for the error "no room for a new `\write`", which occurs when too many macro packages reserve streams to write data to various auxiliary files. It is in principle possible to rewrite packages so that they are less greedy on resources, but that is often unpractical for the end-user. Instead, morewrites hooks at the lowest level (TeX primitives). If I did my job correctly, you simply need to add the line `\usepackage{morewrites}` somewhere near the beginning of your LaTeX file, and the "no room for a new `\write`" error should vanish.

I have tried to make the code as robust as possible, but there may still be bugs lurking as this package has not been tested very thoroughly yet. I thus encourage you to check that references are correct after loading that package: if they are correct without morewrites, but wrong with, please send me a minimal file showing the problem, or post a question on the `tex.stackexchange.com` question and answers website, or the `comp.text.tex` newsgroup.

This package loads the expl3 package, hence the l3kernel bundle needs to be up to date. If Heiko Oberdiek's package atbegshi is available, it will be used.

2 Known deficiencies

Some distributions of TeX allow a quoted syntax for file names with spaces. I haven't yet coded that. A temporary fix is to avoid file names with spaces.

The package code is not very legible, and definitely uses too many `:D` control sequences, whose name means "do not use". The author does not see a way to avoid using primitives in this package, since hooking into the primitives `\immediate`, `\write`, *etc.* requires having a very strong control on what every command does. *Do not take this package as an example of how to code with expl3; go and see Joseph Wright's siunitx instead.*

In particular, I'd like to document better how and when `\newlinechar` and `\endlinechar` are set and used, to make sure that this is done correctly.

3 morewrites implementation

<*package>

```
\ProvidesExplPackage
  {morewrites} {2012/07/10} {0.2} {Always room for a new write}
\RequirePackage{expl3, primargs}
```

⟨@@ = morewrites⟩

3.1 Overview of relevant TeX facts

The aim of the morewrites package is to lift TeX's restriction of only having 16 files open for writing at the same time. We must thus patch 4 primitives, `\openout`, `\write`, `\closeout` and `\immediate`, and the `\newwrite` macro, defined by LaTeX (and plain

TEX). Each of those commands must be made to accept numbers outside the range $[0, 15]$. Let us review the syntax of the various commands we need to alter (see Chapter 24 of the TEXbook).

We start with the three "actions". TEX searches the path for a file with a name given by ⟨*file name*⟩. If found, this file is opened in the writing stream ⟨*integer*⟩, which must be a number in the range $[0, 15]$. TEX expands the ⟨*general text*⟩ as for an `x`-type expansion, with the caveat that macro parameter characters do not need to be doubled; converts the result to a string, and writes it in the writing stream ⟨*integer*⟩. If the writing stream ⟨*integer*⟩ is open (in particular it must be in the range $[0, 15]$), then this writes to the corresponding file. Otherwise, if the ⟨*integer*⟩ is negative, the text is written to the log file, and a non-negative ⟨*integer*⟩ writes to the terminal. One exception: if the ⟨*integer*⟩ is 18, the text is sent to a shell to be run as shell code. If the writing stream ⟨*integer*⟩ is open, it is closed. Otherwise, if the ⟨*integer*⟩ is not in the range $[0, 15]$ an error may be raised, or nothing happens.

By default, each one of those three "actions" are recorded in a whatsit node in the current list, and will be performed when the box containing the whatsit node is sent to the final `pdf`, *i.e.*, at "shipout" time. In particular, the ⟨*general text*⟩ for the `\write` primitive is expanded at shipout time. This behaviour may be modified by putting `\immediate` before any of the three "actions" to force TEX to perform the action immediately instead of recording it in a whatsit node.

Since the `\openout`, `\write`, and `\closeout` primitives operate at `\shipout` time, we will have to hook into this primitive too. It expects to be followed by a box specification such as `\box`⟨*integer*⟩, or `\hbox{`⟨*material to typeset*⟩`}`.

Finally, the `\newwrite` macro expects one token as its argument, and defines this token (with `\chardef`) to be an integer corresponding to the first available writing stream. We must extend it to let it allocate higher (virtual) write registers.

All of the primitives above perform full expansion of all tokens when looking for their operands. In most cases, only the `\meaning` of tokens encountered in this way matters. Specifically,

- ⟨*integer*⟩ denotes an integer in any form that TEX accepts as the right-hand side of a primitive integer assignment of the form `\count0=`⟨*integer*⟩;
- ⟨*equals*⟩ is an arbitrary (optional) number of explicit or implicit space characters, an optional explicit equal sign of category other, and further (optional) explicit or implicit space characters;
- ⟨*file name*⟩ is an arbitrary sequence of explicit or implicit characters with arbitrary category codes (except active characters, which are expanded before reaching TEX's mouth), ending either with a space character (character code 32, arbitrary non-active category code, explicit or implicit), which is removed, or with a non-expandable token, with some care needed for the case of a `\notexpanded:` expandable token;
- ⟨*filler*⟩ is an arbitrary combination of tokens whose meaning is `\relax` or a character with category code 10;

- ⟨*general text*⟩ is formed of braced tokens, starting with an explicit or implicit begin-group character, and ending with the matching explicit end-group character (both with any character code), with an equal number of explicit begin-group and end-group characters in between: this is precisely the right-hand side of an assignment of the form `\toks0=`⟨*general text*⟩.

3.2 Renaming primitives (again)

`\__morewrites_tex_immediate:w`
`\__morewrites_tex_openout:w`
`\__morewrites_tex_write:w`
`\__morewrites_tex_closeout:w`

First save the output-related primitives.

```
\cs_new_eq:NN \__morewrites_tex_immediate:w \tex_immediate:D
\cs_new_eq:NN \__morewrites_tex_openout:w   \tex_openout:D
\cs_new_eq:NN \__morewrites_tex_write:w     \tex_write:D
\cs_new_eq:NN \__morewrites_tex_closeout:w  \tex_closeout:D
```

(*End definition for* `\__morewrites_tex_immediate:w` *and others. These functions are documented on page* **??**.)

`\__morewrites_tex_shipout:w`

Since the non-`\immediate` output primitives act at `\shipout` time, we need to alter this primitive too.

```
\cs_new_eq:NN \__morewrites_tex_shipout:w   \tex_shipout:D
```

(*End definition for* `\__morewrites_tex_shipout:w` *This function is documented on page* **??**.)

3.3 Variables

`\g__morewrites_late_write_int`

The integer `\g__morewrites_late_write_int` labels the various non-immediate operations in the order in which they appear in the source. We can never reuse a number because there is no way to know if a whatsit was recorded in a box register, which could be reused in a shipped-out box:

```
\vbox_set:Nn \l_my_box
{ \iow_shipout_x:Nn \c_term_iow {⟨text⟩} } \tex_shipout:D \tex_-
copy:D \l_my_box \tex_shipout:D \tex_copy:D \l_my_box
```

will print ⟨*text*⟩ to the terminal twice.

```
\int_new:N \g__morewrites_late_write_int
```

(*End definition for* `\g__morewrites_late_write_int` *This variable is documented on page* **??**.)

`\g__morewrites_iow_prop`

The property list `\g__morewrites_iow_prop` associates a file name to each open stream.

```
\prop_new:N \g__morewrites_iow_prop
```

(*End definition for* `\g__morewrites_iow_prop` *This variable is documented on page* **??**.)

`\g__morewrites_iow`
`\g__morewrites_ior`
`\g__morewrites_tmp_file_tl`

The expansion that `\write` performs is impossible to emulate with anything else than `\write`. We will write on the stream `\g__morewrites_iow` to the file `\g__morewrites_-tmp_file_tl` and read back from it in the stream `\g__morewrites_ior` for things to work properly. Unfortunately, this means that the file is repeatedly opened and closed, leaving a trace of that in the log.

```
\newwrite \g__morewrites_iow
\newread \g__morewrites_ior
```

```
\tl_new:N \g__morewrites_tmp_file_tl
\tl_gset:Nn \g__morewrites_tmp_file_tl { \jobname.mw }
```

(*End definition for* `\g__morewrites_iow`*,* `\g__morewrites_ior`*, and* `\g__morewrites_tmp_file_tl` *These variables are documented on page* ??.)

`\g__morewrites_reserved_iow_clist` Some of the writing streams are already allocated when loading this package, and we let the engine manage them. This variable is a clist because it only contains integers and the main task is to test if a given integer is in the comma list.

```
\clist_new:N \g__morewrites_reserved_iow_clist
\int_step_inline:nnnn {0} {1} { \g__morewrites_iow - 1 }
  { \clist_gput_right:Nn \g__morewrites_reserved_iow_clist {#1} }
\clist_gput_right:Nn \g__morewrites_reserved_iow_clist {18}
```

(*End definition for* `\g__morewrites_reserved_iow_clist` *This variable is documented on page* ??.)

`\g__morewrites_stream_int` An integer holding the ⟨*number*⟩ argument of various primitives, namely a writing stream.

```
\int_new:N \g__morewrites_stream_int
```

(*End definition for* `\g__morewrites_stream_int` *This variable is documented on page* ??.)

`\s__morewrites` A recognizable version of `\scan_stop:`. This is inspired from[1] scan marks (see the l3quark module of LaTeX3), but note that we don't use `\__scan_new:N` directly, since it is internal to LaTeX3.

```
\cs_new_eq:NN \s__morewrites \scan_stop:
```

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

`\l__morewrites_internal_tl` Temporary token list, used for scratch purposes.

```
\tl_new:N  \l__morewrites_internal_tl
```

(*End definition for* `\l__morewrites_internal_tl` *This variable is documented on page* ??.)

3.4 Parsing

`\__morewrites_equals_file_name:N` Most of the parsing for primitive arguments is done using primargs, except for one case we care about: after its ⟨*number*⟩ argument, the `\openout` primitive expects an ⟨*equals*⟩ (optional spaces and `=`) and a ⟨*file name*⟩.

```
\cs_new_protected:Npn \__morewrites_equals_file_name:N #1
  {
    \group_begin:
      \tex_aftergroup:D #1
      \primargs_remove_equals:N \__morewrites_parse_file_name:
  }
\cs_new_protected:Npn \__morewrites_parse_file_name:
  { \primargs_get_file_name:N \group_end: }
```

(*End definition for* `\__morewrites_equals_file_name:N`)

[1] Historically, this might have happened the other way around, since the author of this package is also on the LaTeX3 Team.

3.5 Immediate (writing)

In the context of immediate writing, we can store the text in a token list, and only write it at the corresponding `\closeout` command. We keep track of a property list, `\g__morewrites_iow_prop`, of the writes which are open (from the point of view of the user), with the corresponding file name.

3.5.1 What follows `\immediate`

`\__morewrites_immediate:w`
`\__morewrites_immediate_ii:`
`\__morewrites_immediate_iii:N`
`\__morewrites_immediate_iv:NN`
`\__morewrites_immediate_v:w`

This is a little bit subtle: TEX's `\immediate` primitive raises a flag which is cancelled once TEX sees a non-expandable token. We use primargs's `read_x_token` function to fully expand in the TEX way, then test for `\openout`, `\write`, or `\closeout`. We don't test for the primitives themselves, but rather for a recognizable marker, `\s__morewrites`, equal to `\relax`. If present, replace `morewrites` by `morewrites_immediate` in the csname of the second token after it (it turns out that this is the correct structure).

```
\cs_new_protected_nopar:Npn \__morewrites_immediate:w
  { \primargs_read_x_token:N \__morewrites_immediate_ii: }
\cs_new_protected_nopar:Npn \__morewrites_immediate_ii:
  {
    \token_if_eq_meaning:NNT \g_primargs_token \s__morewrites
      { \__morewrites_immediate_iii:N }
  }
\cs_new_protected:Npn \__morewrites_immediate_iii:N #1
  {
    \tl_if_eq:nnTF { #1 } { \s__morewrites }
      { \__morewrites_immediate_iv:NN }
      { #1 }
  }
\cs_new_protected:Npn \__morewrites_immediate_iv:NN #1 #2
  {
    \exp_args:Nc #1
      {
        \exp_after:wN \__morewrites_immediate_v:w
        \token_to_str:N #2
      }
  }
\use:x
  {
    \cs_new:Npn \exp_not:N \__morewrites_immediate_v:w
        ##1 \tl_to_str:n { __morewrites } { __morewrites_immediate }
  }
```

(*End definition for* `\__morewrites_immediate:w` *This function is documented on page* ??.)

3.5.2 Immediate closeout

`\__morewrites_immediate_closeout_test:n`

When the user requests to close a stream, we look in `\g__morewrites_reserved_iow_clist` to see if it is a reserved stream: in this case, we simply use the primitive.

```
\cs_new_protected:Npn \__morewrites_immediate_closeout_test:n #1
```

```
  {
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \g__morewrites_stream_int }
      { \__morewrites_immediate_closeout_aux: }
  }
```

(*End definition for* `\__morewrites_immediate_closeout_test:n`)

`\__morewrites_immediate_closeout_aux:` We then look in `\g__morewrites_iow_prop` to find the file name corresponding to that stream number. If the stream does not appear as a key in the property list, then it was not open yet, and we do nothing. Otherwise, the key is removed, and we write the collected material to the file.

```
\cs_new_protected_nopar:Npn \__morewrites_immediate_closeout_aux:
  {
    \exp_args:NNV \prop_pop:NnNT \g__morewrites_iow_prop
        \g__morewrites_stream_int \l__morewrites_internal_tl
      {
        \__morewrites_immediate_write_and_close:nn
          { \g__morewrites_stream_int } { \l__morewrites_internal_tl }
      }
  }
```

(*End definition for* `\__morewrites_immediate_closeout_aux:`)

`\__morewrites_immediate_write_and_close:nn` The code to write the material collected so far for a given output ⟨*stream*⟩ is in the token list `\g__morewrites_iow_`⟨*stream*⟩`_tl`. We do this writing in the actual stream `\g__morewrites_iow`, briefly opened and closed on the file `#2`.

```
\cs_new_protected:Npn \__morewrites_immediate_write_and_close:nn #1#2
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_openout:w
      \g__morewrites_iow #2 \scan_stop:
    \group_begin:
      \int_set_eq:NN \tex_newlinechar:D \c_minus_one
      \tl_use:c { g__morewrites_iow_ \int_eval:n {#1} _tl }
      \tl_gclear:c { g__morewrites_iow_ \int_eval:n {#1} _tl }
    \group_end:
    \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \g__morewrites_iow
  }
```

(*End definition for* `\__morewrites_immediate_write_and_close:nn`)

3.5.3 Immediate openout

`\__morewrites_immediate_openout_test:n` Read the stream number. If it is one of the reserved streams, we use the primitive. Otherwise, parse an optional equal sign, followed by the file name.

```
\cs_new_protected:Npn \__morewrites_immediate_openout_test:n #1
  {
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_immediate:w \__morewrites_tex_openout:w \g__morewrites_stream_int }
```

```
      { \__morewrites_equals_file_name:N \__morewrites_immediate_openout_aux:n }
  }
```

(*End definition for* `\__morewrites_immediate_openout_test:n`)

`\__morewrites_immediate_openout_aux:n` When the user requests to open a stream, it might already be open, with another file as its destination. We thus need to first close the stream, writing all that we collected so far to that other file. This has no effect if the stream was not open yet.

We then put the stream and its associated file name in the property list, and empty/create the corresponding token list.

```
\cs_new_protected:Npn \__morewrites_immediate_openout_aux:n #1
  {
    \__morewrites_immediate_closeout_aux:
    \prop_gput:NVn \g__morewrites_iow_prop \g__morewrites_stream_int {#1}
    \tl_gclear_new:c { g__morewrites_iow_ \int_use:N \g__morewrites_stream_int _tl }
  }
```

(*End definition for* `\__morewrites_immediate_openout_aux:n`)

3.5.4 Immediate write

`\__morewrites_immediate_write_test:n` Read the stream number. If it is one of the reserved streams, we use the primitive. Otherwise, parse the text.

```
\cs_new_protected:Npn \__morewrites_immediate_write_test:n #1
  {
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_immediate:w \__morewrites_tex_write:w \g__morewrites_stream_int }
      { \primargs_get_general_text:N \__morewrites_immediate_write_aux:n }
  }
```

(*End definition for* `\__morewrites_immediate_write_test:n`)

`\__morewrites_immediate_write_aux:n` Test whether the stream is allocated or not.

```
\cs_new_protected_nopar:Npn \__morewrites_immediate_write_aux:n
  {
    \prop_if_in:NVTF \g__morewrites_iow_prop \g__morewrites_stream_int
      { \__morewrites_immediate_write_open:n }
      { \__morewrites_immediate_write_closed:n }
  }
```

(*End definition for* `\__morewrites_immediate_write_aux:n`)

`\__morewrites_immediate_write_closed:n` If the stream `\g__morewrites_stream_int` is not allocated, then write either to the terminal or only to the log file, depending on the sign.

```
\cs_new_protected:Npn \__morewrites_immediate_write_closed:n #1
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_write:w
      \if_num:w \g__morewrites_stream_int < \c_zero
        -1
      \else:
        16
```

```
      \fi:
      {#1}
  }
```

(*End definition for* `\__morewrites_immediate_write_closed:n`)

`\__morewrites_immediate_write_open:n`
`\__morewrites_immediate_write_readlines_loop:`

Only `\write` itself can emulate how `\write` expands tokens, because `#` don't have to be doubled, and because the `\newlinechar` has to be changed to new lines. Hence, we start by writing `#1` to a file, yielding some lines. The lines are then read one at a time using ε-TEX's `\readline` with `\endlinechar` set to -1 to avoid spurious characters. Each line becomes a `\immediate \write` statement added to the token list `\g__morewrites_iow_`⟨*stream*⟩`_tl`. This token list will be called when it is time to actually write to the file. At that time, `\newlinechar` will be -1, so that writing each line will produce no extra line.

```
\cs_new_protected:Npn \__morewrites_immediate_write_open:n #1
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_openout:w \g__morewrites_iow
      \g__morewrites_tmp_file_tl \scan_stop:
    \__morewrites_tex_immediate:w \__morewrites_tex_write:w \g__morewrites_iow {#1}
    \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \g__morewrites_iow
    \group_begin:
      \int_set_eq:NN \tex_endlinechar:D \c_minus_one
      \tex_openin:D \g__morewrites_ior \g__morewrites_tmp_file_tl \scan_stop:
      \__morewrites_immediate_write_readlines_loop:
      \tex_closein:D \g__morewrites_ior
    \group_end:
  }
\cs_new_protected_nopar:Npn \__morewrites_immediate_write_readlines_loop:
  {
    \etex_readline:D \g__morewrites_ior to \l__morewrites_internal_tl
    \ior_if_eof:NF \g__morewrites_ior
      {
        \tl_gput_right:cx
          { g__morewrites_iow_ \int_use:N \g__morewrites_stream_int _tl }
          {
            \__morewrites_tex_immediate:w \__morewrites_tex_write:w \g__morewrites_iow
              { \l__morewrites_internal_tl }
          }
        \__morewrites_immediate_write_readlines_loop:
      }
  }
```

(*End definition for* `\__morewrites_immediate_write_open:n` *This function is documented on page* **??**.)

3.6 Non-immediate writing

This is trickier, because the expansion of the text for a non-immediate `\write` takes place immediately after the page containing it is shipped out. We store each non-immediate `\openout`, `\write`, or `\closeout` without expansion in separate token lists `\g__morewrites_late_write_`⟨*stream*⟩`_tl` to be used later, and instead write `‘(`⟨*stream*⟩`)`

to a file (including the strange delimiters). After each shipout, we can read the file to see which output operations we need to perform, and in what order.

3.6.1 Replacement for primitives

__morewrites_late:n

Store the action to be done at shipout in a token list, and non-immediately write the label `\g__morewrites_late_write_int` of the output operation to the temporary file. Here, `#1` holds an assignment similar to the lines above it, and `#2` holds the relevant immediate action to be performed after shipout.

```
\cs_new_protected:Npn \__morewrites_late:n #1
  {
    \int_gincr:N \g__morewrites_late_write_int
    \tl_const:cx
      {
        c__morewrites_late_write_
        \int_use:N \g__morewrites_late_write_int
        _tl
      }
      {
        \int_gset:Nn \exp_not:N \g__morewrites_stream_int
          { \exp_not:V \g__morewrites_stream_int }
        \exp_not:n {#1}
      }
    \exp_args:NNx \__morewrites_tex_write:w \g__morewrites_iow
      { `( \int_use:N \g__morewrites_late_write_int ) }
  }
```

(*End definition for* `\__morewrites_late:n`)

__morewrites_openout:w
__morewrites_openout_test:n
__morewrites_openout_aux:n

`\openout` tests if the number to come is among reserved streams. If it is, use the primitive, otherwise, parse a file name.

```
\cs_new_protected_nopar:Npn \__morewrites_openout:w
  { \s__morewrites \primargs_get_number:N \__morewrites_openout_test:n }
\cs_new_protected:Npn \__morewrites_openout_test:n #1
  {
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_openout:w \g__morewrites_stream_int }
      { \__morewrites_equals_file_name:N \__morewrites_openout_aux:n }
  }
\cs_new_protected:Npn \__morewrites_openout_aux:n #1
  { \__morewrites_late:n { \__morewrites_immediate_openout_aux:n {#1} } }
```

(*End definition for* `\__morewrites_openout:w` *This function is documented on page* **??**.)

__morewrites_write:w
__morewrites_write_test:n
__morewrites_write_aux:n

Same idea for `\write`, except that we parse a text.

```
\cs_new_protected_nopar:Npn \__morewrites_write:w
  { \s__morewrites \primargs_get_number:N \__morewrites_write_test:n }
\cs_new_protected:Npn \__morewrites_write_test:n #1
  {
```

```
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_write:w \g__morewrites_stream_int }
      { \primargs_get_general_text:N \__morewrites_write_aux:n }
  }
\cs_new_protected:Npn \__morewrites_write_aux:n #1
  { \__morewrites_late:n { \__morewrites_immediate_write_aux:n {#1} } }
```

(*End definition for* `\__morewrites_write:w` *This function is documented on page* ??.)

`\__morewrites_closeout:w` Same idea for `\closeout`, and we don't need to parse anything else than the number.

`\__morewrites_closeout_test:n` `\__morewrites_closeout_aux:`

```
\cs_new_protected_nopar:Npn \__morewrites_closeout:w
  { \s__morewrites \primargs_get_number:N \__morewrites_closeout_test:n }
\cs_new_protected:Npn \__morewrites_closeout_test:n #1
  {
    \int_gset:Nn \g__morewrites_stream_int {#1}
    \clist_if_in:NnTF \g__morewrites_reserved_iow_clist {#1}
      { \__morewrites_tex_closeout:w \g__morewrites_stream_int }
      { \__morewrites_closeout_aux: }
  }
\cs_new_protected_nopar:Npn \__morewrites_closeout_aux:
  { \__morewrites_late:n { \__morewrites_immediate_closeout_aux: } }
```

(*End definition for* `\__morewrites_closeout:w` *This function is documented on page* ??.)

3.6.2 Shipout business

`\__morewrites_before_shipout:` Immediately before the shipout, we must open the writing stream `\g__morewrites_iow`. Each delayed output operation has been replaced by `\write \g__morewrites_iow {‘(`⟨*operation number*⟩`}`. The delimiters we chose to put around numbers must be at least two distinct characters on the left (then `\tex_newlinechar:D` cannot be equal to the delimiter), and at least one non-digit character on the right.

```
\cs_new_protected_nopar:Npn \__morewrites_before_shipout:
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_openout:w \g__morewrites_iow
      \g__morewrites_tmp_file_tl \scan_stop:
  }
```

(*End definition for* `\__morewrites_before_shipout:`)

`\__morewrites_after_shipout:` `\__morewrites_after_shipout_loop:ww` Immediately after all the `\write`s are performed, close the file, then read the file with `\endlinechar` set to `\newlinechar`[2] to get exactly the original characters that have been written, possibly with extra characters between `‘(...)` groups. The file is then read with all the appropriate category codes set up (no other character can appear in the file). The looping auxiliary `\__morewrites_after_shipout_loop:ww` extract the ⟨*operation*⟩ numbers from the file, and makes a token list out of those. This token list is then used

[2] Note that the `\newlinechar` used by `\write`s at `\shipout` time are those in effect when the page is shipped out, *i.e.*, just after the closing brace of the `\shipout` construction, which is exactly where we have added this hook.

in a mapping function to perform the appropriate `\write` operations. Note that those operations may reuse the file, so we have to fully parse the file before moving on.

```
\cs_new_protected_nopar:Npn \__morewrites_after_shipout:
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \g__morewrites_iow
    \group_begin:
      \int_set_eq:NN \tex_endlinechar:D \tex_newlinechar:D
      \char_set_catcode_other:n { \tex_endlinechar:D }
      \tl_map_inline:nn { ‘(0123456789) }
        { \char_set_catcode_other:n {‘##1} }
      \etex_everyeof:D { ‘() \exp_not:N }
      \exp_args:NNx
    \group_end:
    \tl_map_inline:nn
      {
        \exp_after:wN \__morewrites_after_shipout_loop:ww
        \tex_input:D \g__morewrites_tmp_file_tl \c_space_tl %^^A bug?
      }
      { \tl_use:c { c__morewrites_late_write_ ##1 _tl } }
  }
\cs_new:Npn \__morewrites_after_shipout_loop:ww #1 ‘( #2 )
  {
    \tl_if_empty:nF {#2}
      {
        {#2}
        \__morewrites_after_shipout_loop:ww
      }
  }
```

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

`\shipout`
`\__morewrites_shipout:w`
`\g__morewrites_group_level_int`
`\g__morewrites_shipout_box`

If atbegshi is available, patch it by adding `\__morewrites_before_shipout:` and `\__morewrites_after_shipout:` at the right place: the two transformations are needed to cover several versions of the package. Otherwise, redefine `\shipout` to add a hook (see Heiko's atbegshi for details).

```
\IfFileExists{atbegshi.sty}
  {
    \RequirePackage{atbegshi}
    \tl_replace_once:Nnn \AtBegShi@Output
      { \AtBegShi@OrgShipout \box \AtBeginShipoutBox }
      {
        \__morewrites_before_shipout:
        \AtBegShi@OrgShipout \box \AtBeginShipoutBox
        \__morewrites_after_shipout:
      }
    \tl_replace_once:Nnn \AtBegShi@Output
      { \AtBeginShipoutOriginalShipout \box \AtBeginShipoutBox }
      {
        \__morewrites_before_shipout:
        \AtBeginShipoutOriginalShipout \box \AtBeginShipoutBox
```

```
        \__morewrites_after_shipout:
      }
  }
  {
    \int_new:N \g__morewrites_group_level_int
    \box_new:N \g__morewrites_shipout_box
    \cs_new_protected_nopar:Npn \__morewrites_shipout:w
      {
        \int_gset_eq:NN \g__morewrites_group_level_int \etex_currentgrouplevel:D
        \tex_afterassignment:D \__morewrites_shipout_i:
        \tex_global:D \tex_setbox:D \g__morewrites_shipout_box
      }
    \cs_new_protected_nopar:Npn \__morewrites_shipout_i:
      {
        \int_compare:nNnTF { \g__morewrites_group_level_int }
                           = { \etex_currentgrouplevel:D }
          { \__morewrites_shipout_ii: }
          { \tex_aftergroup:D \__morewrites_shipout_ii: }
      }
    \cs_new_protected_nopar:Npn \__morewrites_shipout_ii:
      {
        \__morewrites_before_shipout:
        \__morewrites_tex_shipout:w \tex_box:D \g__morewrites_shipout_box
        \__morewrites_after_shipout:
      }
    \AtBeginDocument { \cs_gset_eq:NN \shipout \__morewrites_shipout:w }
  }
```

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

3.7 Hook at the very end

`\g__morewrites_at_end_int`

At the end of the run, we try very hard to put some material at the `\@@end`. This integer controls how many times to call `\__morewrites_close_all_at_end:w`, to avoid infinite loops in case two packages compete for that last place.

```
\int_new:N \g__morewrites_at_end_int
\int_gset:Nn \g__morewrites_at_end_int { 10 }
```

(*End definition for* `\g__morewrites_at_end_int` *This variable is documented on page* **??**.)

`\__morewrites_close_all:`

At the end of the document, close all the files.

```
\cs_new_protected_nopar:Npn \__morewrites_close_all:
  {
    \prop_map_function:NN \g__morewrites_iow_prop
      \__morewrites_immediate_write_and_close:nn
    \prop_gclear:N \g__morewrites_iow_prop
  }
```

(*End definition for* `\__morewrites_close_all:`)

__morewrites_close_all_at_end:w This pushes its first argument to the very end of the LaTeX run, recursively (at most 10 times, initial value of \g__morewrites_at_end_int), just in case some other code adds things there.

```
\cs_set:Npn \__morewrites_tmp:w #1
  {
    \cs_new_protected:Npn \__morewrites_close_all_at_end:w ##1 #1
      {
        \int_gdecr:N \g__morewrites_at_end_int
        \int_compare:nNnTF \g__morewrites_at_end_int > \c_zero
          {
            \tl_if_empty:nTF {##1}
              { ##1 \__morewrites_close_all: }
              { ##1 \__morewrites_close_all_at_end:w }
          }
          { \__morewrites_close_all: ##1 }
        #1
      }
  }
\exp_args:Nc \__morewrites_tmp:w { @ @ end }
\AtEndDocument { \__morewrites_close_all_at_end:w }
```

(*End definition for* __morewrites_close_all_at_end:w)

3.8 Modified \newwrite

\g__morewrites_alloc_int The counter that LaTeX 2_ε uses to allocate \write registers.

```
\tex_countdef:D \g__morewrites_alloc_int 17 \scan_stop:
```

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

\newwrite We need to allow \newwrite to allocate more than 16 writes, but beware that 18 is reserved, and that packages might expect 16 or 17 to write to the terminal. So instead skip until 20, to be on the safe side.

```
\cs_new:Npn \__morewrites_newwrite:N #1
  {
    \int_gincr:N \g__morewrites_alloc_int
    \if_num:w \g__morewrites_alloc_int = \c_sixteen
      \int_gset:Nn \g__morewrites_alloc_int { 20 }
    \fi:
    \int_set_eq:NN \allocationnumber \g__morewrites_alloc_int
    \cs_undefine:N #1
    \int_const:Nn #1 { \allocationnumber }
    \wlog
      {
        \token_to_str:N #1
        = \token_to_str:N \write \int_use:N \allocationnumber
      }
  }
```

(*End definition for* \newwrite)

3.9 Redefining the "normal" control sequences

\immediate \openout \write \closeout \newwrite

`\shipout` has been redefined earlier.

```
\cs_gset_eq:NN \immediate \__morewrites_immediate:w
\cs_gset_eq:NN \openout   \__morewrites_openout:w
\cs_gset_eq:NN \write     \__morewrites_write:w
\cs_gset_eq:NN \closeout  \__morewrites_closeout:w
\cs_gset_eq:NN \newwrite  \__morewrites_newwrite:N
```

(*End definition for* `\immediate` *and others.*)

</package>