The morewrites package: Always room for a new \write

Bruno Le Floch

2017/04/10

Contents

1 morewrites documentation

This LaTeX package is a solution for the errors

- "no room for a new \write",
- "no room for a new \read",

which occur when a document reserves too many streams to write data to various auxiliary files or read from them. It is in principle possible to rewrite other 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).

Simply add the line \usepackage{morewrites} near the beginning of your LaTeX file's preamble: the "no room for a new \write/\read" error should vanish. If it does not, please contact me so that I can correct the problem. This can be done by posting a question on the tex.stackexchange.com question and answers website, logging an issue on GitHub (https://github.com/blefloch/latex-morewrites), or emailing me a minimal file showing the problem.

Notes.

- This package loads the expl3 package, hence the l3kernel bundle needs to be up to date.
- This package uses an auxiliary file, ⟨*job name*⟩.mw, which can safely be deleted. Versions from 2015 and later will only use the auxiliary file if it is originally empty, to avoid destroying data (such as .mw files used by Maple). This means that .mw files generated by versions before 2015 should be deleted by hand.
- LuaTeX allows 128 \write streams, so this package does nothing (with a warning) when used with LuaTeX.

1.1 Commands defined or altered by morewrites

`\morewritessetup`

New: 2014-07-26

`\morewritessetup {⟨key-value list⟩}`

Sets the options described by the ⟨*key–value list*⟩.

`allocate`

New: 2017-04-10

`\morewritessetup { allocate = ⟨integer⟩ }`

Sets to (at least) ⟨*integer*⟩ the number of `\write` streams allocated to the inner workings of morewrites. By default this is zero but increasing this value may speed up morewrites.

`file`

New: 2014-07-26
Updated: 2015-08-01

`\morewritessetup { file = ⟨file name⟩ }`

Sets (globally) the name of the file which will be used by internal processes of morewrites. The file name is `\jobname.mw` by default (technically, `\c_sys_jobname_str.mw`). Contrarily to earlier versions of morewrites non-empty files will not be overwritten; this design choice may lead to unwanted `.mw` files remaining.

`\newread`
`\newwrite`

Updated: 2015-08-01

These macros are redefined by morewrites. Since morewrites allows more than 16 read/write streams, it removes the corresponding restrictions in `\newread` and `\newwrite`.

TEXhackers note: The revised `\newread` and `\newwrite` allocate stream numbers starting at 19. This might break some code that expects stream numbers to be less than 16.

`\immediate`

Updated: 2015-08-01

This primitive is altered by morewrites, to detect a following `\write` or `\openout` or `\closeout` and perform the appropriate action.

`\openin`
`\read`
`\readline`
`\closein`
`\openout`
`\write`
`\closeout`

Updated: 2015-08-01

These seven primitives are altered by morewrites so that they accept stream numbers outside the normal range $[0, 15]$ and open/read/write/close files as appropriate.

TEXhackers note: System calls using `\write18` are detected and forwarded to the engine.

`\shipout`

This primitive is altered by morewrites to ensure that delayed `\openout`, `\write` and `\closeout` commands are performed at `\shipout` time, and in the correct order.

1.2 Known deficiencies and open questions

See the bug tracker https://github.com/blefloch/latex-morewrites/issues/ for a list of issues with morewrites.

The package code is not good expl3 code. *Do not take this package as an example of how to code with expl3; go and see Joseph Wright's siunitx instead.* It uses `\...:D` primitives directly (the `:D` stands for "do not use"). This is unavoidable in order to hook into the primitives `\immediate`, `\write`, *etc.* and to keep a very strong control on what every command does.

2 morewrites implementation

<*package>

```
\RequirePackage {expl3} [2017/03/18]
\RequirePackage {primargs} [2017/04/10]
\ProvidesExplPackage
  {morewrites} {2017/04/10} {} {Always room for a new write}
```

Quit early under LuaTEX.

```
\sys_if_engine_luatex:T
  {
    \cs_new_protected:Npn \morewritessetup #1 { }
    \msg_new:nnn { morewrites } { luatex }
      { The~morewrites~package~is~unnecessary~in~LuaTeX. }
    \msg_warning:nn { morewrites } { luatex }
    \tex_endinput:D
  }%
```

⟨@@=morewrites⟩

2.1 Overview of relevant TEX facts

The aim of the morewrites package is to lift TEX's restriction of only having 16 files open for reading at the same time, and the same restriction for writing. This requires patching the primitives `\openin`, `\read`, `\readline`, `\closein`, `\immediate`, `\openout`, `\write`, `\closeout`, and `\shipout`, and the macros `\newread` and `\newwrite` present in plain TEX and LATEX 2_ε.

The morewrites package should be loaded as early as possible, so that any package loaded later uses the redefined macros instead of the primitives. However, the format (plain TEX or LATEX 2_ε) and the expl3 programming language are always loaded before morewrites, and their interaction must be carefully monitored.

Henceforth, "TEX stream" will refer to stream numbers in the range $[0, 15]$ provided to TEX's read/write primitives, while "user stream" will denote stream numbers in $[0, 15] \cup [19, \infty)$ manipulated by the redefined `\openin`, `\read`, `\readline`, `\closein`, `\openout`, `\write`, `\closeout`, `\newread`, and `\newwrite`. A user stream in $[0, 15]$ (reserved by LATEX 2_ε or allocated by expl3) is mapped to the same TEX stream number, while a user stream in $[19, \infty)$ is mapped to a TEX stream according to the property lists (with integer keys and values) `\l__morewrites_read_prop` and `\l__morewrites_write_prop`. Stream numbers 16, 17 and 18 are unused because `\write16` is often used to write to the terminal, and `\write18` sends its argument to a shell.

The primitives `\openin`, `\read`, `\readline`, `\closein`, `\openout`, `\write`, and `\closeout` expect to be followed by an ⟨*integer*⟩, normally in the range $[0, 15]$, then some further arguments.

`\openin` ⟨*integer*⟩ ⟨*equals*⟩ ⟨*file name*⟩
`\read` ⟨*integer*⟩ `to` ⟨*control sequence*⟩
`\readline` ⟨*integer*⟩ `to` ⟨*control sequence*⟩
`\closein` ⟨*integer*⟩
`\openout` ⟨*integer*⟩ ⟨*equals*⟩ ⟨*file name*⟩
`\write` ⟨*integer*⟩ ⟨*filler*⟩ ⟨*general text*⟩
`\closeout` ⟨*integer*⟩

All of the primitives above perform full expansion of all tokens when looking for their operands.

- ⟨*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 whose category code is 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*⟩.

The morewrites package redefines these seven control sequences to expect a user stream number rather than a TEX stream number as the ⟨*integer*⟩, then map such a user stream to a TEX stream to call the primitive with the appropriate argument. The primitive `\immediate` must also be redefined to detect `\openout`, `\write`, and `\closeout` and make them immediate, while still working with other primitives that can be made immediate. Finally, `\newread` and `\newwrite` must be patched to allocate stream numbers beyond 15.

A few comments on the behaviour of primitives concerning the ⟨*integer*⟩ (TEX stream). The primitives `\openin` and `\openout` trigger errors if the ⟨*integer*⟩ is not in $[0, 15]$. The primitives `\read` and `\readline` prompt the user for input if the TEX stream is closed or beyond $[0, 15]$, with no explicit prompt if the stream number is negative. The primitive `\write` outputs to the log if the ⟨*integer*⟩ is negative, and to the terminal if the TEX stream is closed or greater than 15, with the exception of `\write18` which runs code in a shell. The primitives `\closein` and `\closeout` trigger errors if the ⟨*integer*⟩ is not in $[0, 15]$ and silently do nothing if the TEX stream is not open, with the exception of `\closeout18` which causes a segfault at least in some versions.

By default, `\openout`, `\write` and `\closeout` 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 these three primitives 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, for instance `\box`⟨*integer*⟩ or `\hbox{`⟨*material to typeset*⟩`}`.

Finally, the `\newread` and `\newwrite` macros expect one token as their argument, and define this token (with `\chardef`) to be an integer corresponding to the first available (TEX) read/write stream. This must be extended to allocate higher (user) streams.

2.2 Preliminaries

2.2.1 Copying some commands

__morewrites_tex_immediate:w
__morewrites_tex_openout:w
__morewrites_tex_write:w
__morewrites_tex_closeout:w
__morewrites_tex_openin:w
__morewrites_tex_read:w
__morewrites_tex_readline:w
__morewrites_tex_closein:w

Aliases for the read- and write-related primitives, to avoid having `:D` throughout the code.

```
\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
\cs_new_eq:NN \__morewrites_tex_openin:w    \tex_openin:D
\cs_new_eq:NN \__morewrites_tex_read:w      \tex_read:D
\cs_new_eq:NN \__morewrites_tex_readline:w  \etex_readline:D
\cs_new_eq:NN \__morewrites_tex_closein:w   \tex_closein:D
```

(*End definition for* `\__morewrites_tex_immediate:w` *and others.*)

__morewrites_tex_newread:N
__morewrites_tex_newwrite:N

Copy `\newread` and `\newwrite` but making sure that they are not `\outer`. These copies will not be affected by redefinitions of `\newread` and `\newwrite` later on.

```
\exp_args:NNf \cs_new_protected:Npn \__morewrites_tex_newread:N
  { \exp_args:NNc \exp_after:wN \exp_stop_f: { newread } }
\exp_args:NNf \cs_new_protected:Npn \__morewrites_tex_newwrite:N
  { \exp_args:NNc \exp_after:wN \exp_stop_f: { newwrite } }
```

(*End definition for* `\__morewrites_tex_newread:N` *and* `\__morewrites_tex_newwrite:N`.)

2.2.2 Variants

\prop_gpop:NVNT
\prop_gput:NVx
\tl_gput_right:Nv

We need these variants.

```
\cs_generate_variant:Nn \prop_gpop:NnNT { NV }
\cs_generate_variant:Nn \prop_gput:Nnn { NVx }
\cs_generate_variant:Nn \tl_gput_right:Nn { Nv }
```

(*End definition for* `\prop_gpop:NVNT`, `\prop_gput:NVx`, *and* `\tl_gput_right:Nv`.)

2.2.3 Variables

\l__morewrites_internal_tl

Used for temporary scratch purposes.

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

(*End definition for* `\l__morewrites_internal_tl`.)

__morewrites_tmp:w

Used for temporary definitions.

```
\cs_new_eq:NN \__morewrites_tmp:w ?
```

(*End definition for* `\__morewrites_tmp:w`.)

\g__morewrites_later_int

The integer `\g__morewrites_later_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⟩} } \shipout \copy \l_my_box
\shipout \copy \l_my_box
```

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

```
\int_new:N \g__morewrites_later_int
```

(*End definition for* `\g__morewrites_later_int`.)

`\g__morewrites_read_seq` `\g__morewrites_write_seq`

Keep track of TEX stream numbers managed by morewrites that are currently not in use as user streams.

```
\seq_new:N \g__morewrites_read_seq
\seq_new:N \g__morewrites_write_seq
```

(*End definition for* `\g__morewrites_read_seq` *and* `\g__morewrites_write_seq`.)

`\g__morewrites_read_prop` `\g__morewrites_write_prop`

Map user streams to TEX streams.

```
\prop_new:N \g__morewrites_read_prop
\prop_new:N \g__morewrites_write_prop
```

(*End definition for* `\g__morewrites_read_prop` *and* `\g__morewrites_write_prop`.)

`\g__morewrites_write_file_prop`

Map user streams with no associated TEX streams to file names.

```
\prop_new:N \g__morewrites_write_file_prop
```

(*End definition for* `\g__morewrites_write_file_prop`.)

`\l__morewrites_code_tl`

Stores the code to run after finding a user stream, in `\__morewrites_get_user:n`.

```
\tl_new:N \l__morewrites_code_tl
```

(*End definition for* `\l__morewrites_code_tl`.)

`\l__morewrites_user_int` `\l__morewrites_tstr_tl`

The user stream number following redefined primitives is stored in `\l__morewrites_user_int` (see `\__morewrites_get_user:N`). The corresponding TEX stream number is eventually stored in `\l__morewrites_tstr_tl` (a token list).

```
\int_new:N \l__morewrites_user_int
\tl_new:N \l__morewrites_tstr_tl
```

(*End definition for* `\l__morewrites_user_int` *and* `\l__morewrites_tstr_tl`.)

`\l__morewrites_tstr_token`

This token is given as an argument to `\__morewrites_tex_newread:N` or `\__morewrites_tex_newwrite:N`.

```
\cs_new_eq:NN \l__morewrites_tstr_token ?
```

(*End definition for* `\l__morewrites_tstr_token`.)

`\s__morewrites`

A recognizable version of `\scan_stop:`. This is inspired by[1] scan marks (see the l3quark module of LATEX3), but `\__scan_new:N` is not used directly, since it is currently internal to LATEX3.

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

(*End definition for* `\s__morewrites`.)

[1] Historically, this might have happened the other way around, since the author of this package is also on the LATEX3 Team.

\g__morewrites_iow
\g__morewrites_ior

The expansion that \write performs is impossible to emulate (in XeTeX at least) 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
```

(*End definition for* \g__morewrites_iow *and* \g__morewrites_ior.)

\g__morewrites_tmp_file_tl
\g__morewrites_tmp_file_bool

Temporary file used to do the correct expansion for each \write. Boolean indicating whether we have already checked that the file can be used by morewrites: before using a file, the morewrites package now checks it is empty, so as to avoid clobbering user data.

```
\tl_new:N \g__morewrites_tmp_file_tl
\bool_new:N \g__morewrites_tmp_file_bool
\bool_gset_false:N \g__morewrites_tmp_file_bool
```

(*End definition for* \g__morewrites_tmp_file_tl *and* \g__morewrites_tmp_file_bool.)

\g__morewrites_group_level_int

The group level when \shipout is called: this is used to distinguish between explicit boxes and box registers.

```
\int_new:N \g__morewrites_group_level_int
```

(*End definition for* \g__morewrites_group_level_int.)

\g__morewrites_shipout_box

The page to be shipped out.

```
\box_new:N \g__morewrites_shipout_box
```

(*End definition for* \g__morewrites_shipout_box.)

2.2.4 Helpers for auxiliary file

__morewrites_set_file:n

Sets \g__morewrites_tmp_file_tl to the given value (initially \c_sys_jobname_str.mw). Mark that the file has not been checked.

```
\cs_new_protected:Npn \__morewrites_set_file:n #1
  {
    \bool_gset_false:N \g__morewrites_tmp_file_bool
    \tl_gset:Nn \g__morewrites_tmp_file_tl {#1}
  }
```

(*End definition for* __morewrites_set_file:n.)

__morewrites_empty_file:n

Empties the file \g__morewrites_tmp_file_tl by opening it and closing it right away. This is used when performing \immediate \openout. It is also used to ensure the file used by morewrites is left empty. We do this every time the auxiliary file is used, in case that run ends with an error mid-document.

```
\cs_new_protected:Npn \__morewrites_empty_file:n #1
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_openout:w
      \g__morewrites_iow = #1 \scan_stop:
    \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w
      \g__morewrites_iow
  }
```

(*End definition for* `\__morewrites_empty_file:n`.)

`\__morewrites_if_file_trivial:nTF` True if the file does not exist, or if it is empty. Only the `TF` variant is defined. We set `\__morewrites_tmp:w` to `\prg_return_true:` or `\prg_return_false:` within the group and use it after cleaning up. The first `eof` test is `true` if the file does not exist. Then we read one line, the second `eof` test is `true` if the file was empty (it is `false` if the file contained anything, even a single space).

```
\prg_new_conditional:Npnn \__morewrites_if_file_trivial:n #1 { TF }
  {
    \group_begin:
      \tex_openin:D \g__morewrites_ior = #1 \scan_stop:
      \if_eof:w \g__morewrites_ior
        \cs_gset_eq:NN \__morewrites_tmp:w \prg_return_true:
      \else:
        \int_set:Nn \tex_endlinechar:D { -1 }
        \etex_readline:D \g__morewrites_ior to \l__morewrites_internal_tl
        \if_eof:w \g__morewrites_ior
          \cs_gset_eq:NN \__morewrites_tmp:w \prg_return_true:
        \else:
          \cs_gset_eq:NN \__morewrites_tmp:w \prg_return_false:
        \fi:
      \fi:
      \tex_closein:D \g__morewrites_ior
    \group_end:
    \__morewrites_tmp:w
  }
```

(*End definition for* `\__morewrites_if_file_trivial:nTF`.)

`\__morewrites_chk_file:` Check that the file `\g__morewrites_tmp_file_tl` does not exist or is blank. If not, try the file name obtained by adding `.mw`. This avoids clobbering files that the user would not want to lose.

```
\cs_new_protected:Npn \__morewrites_chk_file:
  {
    \__morewrites_if_file_trivial:nTF { \g__morewrites_tmp_file_tl }
      { \bool_gset_true:N \g__morewrites_tmp_file_bool }
      {
        \msg_warning:nnxx { morewrites } { file-exists }
          { \g__morewrites_tmp_file_tl }
          { \g__morewrites_tmp_file_tl .mw }
        \tl_gput_right:Nn \g__morewrites_tmp_file_tl { .mw }
        \__morewrites_chk_file:
      }
  }
\msg_new:nnnn { morewrites } { file-exists }
  { File~'#1'~exists,~using~'#2'~instead. }
  {
    The~file~`#1'~exists~and~was~not~created~by~this~version~of~the~
    `morewrites'~package.~Please~move~or~delete~that~file,~or~provide~
    another~file~name~by~adding
    \\ \\
    \iow_indent:n { \iow_char:N\\morewritessetup~{~file~=~other-name~} }
    \\ \\
    to~your~source~file.~In~the~meantime,~the~file~`#2'~will~be~used.
```

```
}
```

(*End definition for* `\__morewrites_chk_file:`.)

2.2.5 Parsing and other helpers

`\__morewrites_equals_file: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:N #1
  {
    \group_begin:
      \tex_aftergroup:D \primargs_get_file_name:N
      \tex_aftergroup:D #1
      \primargs_remove_equals:N \group_end:
  }
```

(*End definition for* `\__morewrites_equals_file:N`.)

`\__morewrites_get_user:n`

primargs commands only take `N`-type arguments, but we often need to find an integer, save it in `\l__morewrites_user_int`, and run some code `#1`. This is analogous to `\primargs_get_number:N`.

```
\cs_new_protected:Npn \__morewrites_get_user:n #1
  {
    \tl_set:Nn \l__morewrites_code_tl {#1}
    \tex_afterassignment:D \l__morewrites_code_tl
    \l__morewrites_user_int =
  }
```

(*End definition for* `\__morewrites_get_user:n`.)

`\__morewrites_user_to_tstr:NTF`

The goal is to go from a user stream `\l__morewrites_user_int` to a TEX stream `\l__morewrites_tstr_tl` (it defaults to the user stream). Streams less than 19 are not managed by morewrites: actual TEX streams in $[0, 15]$; negative for writing to `log` and reading without prompt; 16, 17 for writing to terminal and reading with prompt; 18 for shell escape. Larger stream numbers are looked up in the property list `#1`, either `\g__morewrites_read_prop` or `\g__morewrites_write_prop`. If present, use the corresponding value as the TEX stream, otherwise run the `false` branch.

```
\cs_new_protected:Npn \__morewrites_user_to_tstr:NTF #1
  {
    \tl_set:NV \l__morewrites_tstr_tl \l__morewrites_user_int
    \int_compare:nNnTF { \l__morewrites_user_int } < { 19 }
      { \use_i:nn }
      { \prop_get:NVNTF #1 \l__morewrites_user_int \l__morewrites_tstr_tl }
  }
```

(*End definition for* `\__morewrites_user_to_tstr:NTF`.)

`\l__morewrites_collect_next_int`
`\__morewrites_collect:x`
`\__morewrites_collect_aux:Nn`
`\__morewrites_collect_aux:cf`
`\__morewrites_collect_gput_right:N`
`\__morewrites_collect_gput_right:c`

When encountering very large `\write` statements we may need to collect many lines. This can easily become an $O(n^2)$ task, and here we make sure that it remains around $O(n \log n)$, with a large constant unfortunately. Each of the token lists `\l__morewrites_$k$_tl` is empty or contains 2^k lines. As lines accumulate, they move to

token lists with larger values of k, and eventually all are combined. The integer `\l__morewrites_collect_next_int` is (one plus) the maximal k among non-empty token lists.

```
\int_new:N \l__morewrites_collect_next_int
\cs_new_protected:Npn \__morewrites_collect:x #1
  {
    \tl_set:Nx \l__morewrites_internal_tl {#1}
    \__morewrites_collect_aux:cf { l__morewrites_0_tl } { 1 }
  }
\cs_new_protected:Npn \__morewrites_collect_aux:Nn #1#2
  {
    \int_compare:nNnT {#2} > \l__morewrites_collect_next_int
      {
        \tl_clear_new:N #1
        \int_set:Nn \l__morewrites_collect_next_int {#2}
      }
    \tl_if_empty:NTF #1
      { \tl_set_eq:NN #1 \l__morewrites_internal_tl }
      {
        \tl_put_left:No \l__morewrites_internal_tl {#1}
        \tl_clear:N #1
        \__morewrites_collect_aux:cf { l__morewrites_#2_tl }
          { \int_eval:n { #2 + 1 } }
      }
  }
\cs_generate_variant:Nn \__morewrites_collect_aux:Nn { cf }
\cs_new_protected:Npn \__morewrites_collect_gput_right:N #1
  {
    \int_compare:nNnF \l__morewrites_collect_next_int = 0
      {
        \int_decr:N \l__morewrites_collect_next_int
        \tl_gput_right:Nv #1
          {
            l__morewrites_
            \int_use:N \l__morewrites_collect_next_int
            _tl
          }
        \__morewrites_collect_gput_right:N #1
      }
  }
\cs_generate_variant:Nn \__morewrites_collect_gput_right:N { c }
```

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

`\__morewrites_user_tl_name:n` The name of a global token list variable holding the text of a given user stream.

```
\cs_new:Npn \__morewrites_user_tl_name:n #1
  { g__morewrites_iow_ \int_eval:n {#1} _tl }
```

(*End definition for* `\__morewrites_user_tl_name:n`.)

2.3 Reading

`\__morewrites_openin:w` Set `\l__morewrites_user_int` to a user stream then convert it to a TEX stream `\l__morewrites_tstr_tl` and call the primitive. If the user stream is closed (not associated

to any TEX stream), the `false` branch of `\__morewrites_user_to_tstr:NTF` is taken. Then get a stream from `\g__morewrites_read_seq` (TEX streams managed by morewrites but not in use) or use the copy of `\newread` as a fallback. Store the new mapping from user stream to TEX stream into `\g__morewrites_read_prop`.

```
\cs_new_protected:Npn \__morewrites_openin:w
  {
    \__morewrites_get_user:n
      {
        \__morewrites_user_to_tstr:NTF \g__morewrites_read_prop { }
          {
            \seq_pop:NNF \g__morewrites_read_seq \l__morewrites_tstr_tl
              {
                \__morewrites_tex_newread:N \l__morewrites_tstr_token
                \tl_set:NV \l__morewrites_tstr_tl \l__morewrites_tstr_token
              }
            \prop_gput:NVV \g__morewrites_read_prop \l__morewrites_user_int \l__morewrites_tst
          }
        \__morewrites_tex_openin:w \l__morewrites_tstr_tl \exp_stop_f:
      }
  }
```

(*End definition for* `\__morewrites_openin:w`.)

`\__morewrites_read:w`
`\__morewrites_readline:w`

Set `\l__morewrites_user_int` to a user stream and convert it to a TEX stream `\l__morewrites_tstr_tl` then call the primitive. Nothing needs to be done for streams not managed by morewrites or for user streams that do not correspond to any TEX stream, as the primitive will simply read user input from the terminal as TEX would do.

```
\cs_new_protected:Npn \__morewrites_read:w
  {
    \__morewrites_get_user:n
      {
        \__morewrites_user_to_tstr:NTF \g__morewrites_read_prop { } { }
        \__morewrites_tex_read:w \l__morewrites_tstr_tl \exp_stop_f:
      }
  }
\cs_new_protected:Npn \__morewrites_readline:w
  {
    \__morewrites_get_user:n
      {
        \__morewrites_user_to_tstr:NTF \g__morewrites_read_prop { } { }
        \__morewrites_tex_readline:w \l__morewrites_tstr_tl \exp_stop_f:
      }
  }
```

(*End definition for* `\__morewrites_read:w` *and* `\__morewrites_readline:w`.)

`\__morewrites_closein:w`

There is slightly more work than for `read` and `readline` because closing a user stream managed by morewrites means removing it from the mapping and putting it back in the list of available streams. To avoid useless work, only do this if there was indeed a non-trivial mapping between user and TEX stream.

```
\cs_new_protected:Npn \__morewrites_closein:w
  {
    \__morewrites_get_user:n
```

```
    {
      \__morewrites_user_to_tstr:NTF \g__morewrites_read_prop { } { }
      \int_compare:nNnF { \l__morewrites_tstr_tl } = { \l__morewrites_user_int }
        {
          \prop_gremove:NV \g__morewrites_read_prop \l__morewrites_user_int
          \seq_gput_left:NV \g__morewrites_read_seq \l__morewrites_tstr_tl
        }
      \__morewrites_tex_closein:w \l__morewrites_tstr_tl \exp_stop_f:
    }
}
```

(*End definition for* `\__morewrites_closein:w`.)

2.4 Writing

Writing is much harder than reading for several reasons.

Contrarily to reading, it is possible to hold on to material while a file is being written and only write it in one go once the file closes, to avoid using a stream throughout. At any given time, each user stream may point to an open TEX stream, given in `\g__morewrites_write_prop` (like we did for reading), or may point to a token list that will eventually be written to a file whose file name is stored in `\g__morewrites_write_file_prop`, or may be closed.

When a user stream points to a token list rather than a TEX stream, any material to be written must be written to our temporary file and read back in to apply the same expansion as `\write` does.

Another difficulty is that users may mix immediate and non-immediate operations. The biggest difficulty comes from the possibility of copying boxes containing delayed actions. If we ever produced a whatsit `\write`⟨*number*⟩`{`⟨*text*⟩`}` then the TEX stream ⟨*number*⟩ would have to be reserved forever, as as copies of the box containing this delayed actions may be shipped out at any later point in the document.

Each delayed action is thus saved in a separate numbered token list and `\write\g__morewrites_iow{`⟨*number*⟩`}` is inserted instead of the delayed action. At each `\shipout`, the stream `\g__morewrites_iow` is opened, to catch the ⟨*number*⟩ of each action that should be performed at this `\shipout`.

2.4.1 Redefining `\immediate`

To accomodate the `\immediate` primitive, our versions of `\openout`, `\write` and `\closeout` will take the form

`\s__morewrites \use_i:nn` `{`⟨*code for delayed action*⟩`}`
`{`⟨*code for immediate action*⟩`}`
⟨*further code*⟩

The leading `\s__morewrites` allows the redefined `\immediate` to detect these redefined primitives, and to run the ⟨*code for immediate action*⟩ instead of the ⟨*code for delayed action*⟩ which is run by default. In both cases, any ⟨*further code*⟩ is run.

`\__morewrites_immediate:w`
`\__morewrites_immediate_auxii:`
`\__morewrites_immediate_auxiii:N`

TEX's `\immediate` primitive raises a flag which is cancelled after TEX sees a non-expandable token. We use `\primargs_read_x_token:N` to find the next non-expandable token then test for `\openout`, `\write`, and `\closeout`. More precisely we test for the marker `\s__morewrites` and run the appropriate code as described above. Otherwise

we call the primitive, for cases where the next token is `\pdfobj` or similar. This code performs too much expansion for some nonsensical uses of `\noexpand` after `\immediate`.

```
\cs_new_protected:Npn \__morewrites_immediate:w
  { \primargs_read_x_token:N \__morewrites_immediate_auxii: }
\cs_new_protected:Npn \__morewrites_immediate_auxii:
  {
    \token_if_eq_meaning:NNTF \g_primargs_token \s__morewrites
      { \__morewrites_immediate_auxiii:N }
      { \__morewrites_tex_immediate:w }
  }
\cs_new_protected:Npn \__morewrites_immediate_auxiii:N #1
  { \str_if_eq:nnTF { #1 } { \s__morewrites } { \use_iii:nnn } { #1 } }
```

(*End definition for* `\__morewrites_immediate:w`*,* `\__morewrites_immediate_auxii:`*, and* `\__morewrites_immediate_auxiii:N`*.*)

2.4.2 Immediate actions

The `\openout`, `\write`, and `\closeout` primitive can be either delayed or immediate. In all cases they begin by looking for a user stream. Here, we implement the immediate versions only.

`\__morewrites_closeout:w`
`\__morewrites_closeout_now:`
`\__morewrites_closeout_now:nn`

In the immediate case `\__morewrites_closeout_now:`, there are three cases. The stream may point to a TEX stream, in which case it is closed, removed from `\g__morewrites_write_prop`, and put back in the list of usable streams. The stream may point to a token list, in which case that token list should be written to the appropriate file. The stream may be closed, in which case nothing happens. The auxiliary `\__morewrites_closeout_now:nn` writes the material collected so far for a given user stream `#1` to the file `#2`. This uses the TEX stream `\g__morewrites_iow`. The token list consists of multiple `\immediate \write \g__morewrites_iow {`⟨*text*⟩`}` statements because that is the only safe way to obtain new lines. We do not remove the stream/file pair from `\g__morewrites_write_file_prop`.

```
\cs_new_protected:Npn \__morewrites_closeout:w
  {
    \s__morewrites
    \use_i:nn
      { \__morewrites_get_user:n { \__morewrites_closeout_later: } }
      { \__morewrites_get_user:n { \__morewrites_closeout_now: } }
  }
\cs_new_protected:Npn \__morewrites_closeout_now:
  {
    \__morewrites_user_to_tstr:NTF \g__morewrites_write_prop
      {
        \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \l__morewrites_tstr_tl \exp
        \int_compare:nNnF { \l__morewrites_tstr_tl } = { \l__morewrites_user_int }
          {
            \prop_gremove:NV \g__morewrites_write_prop \l__morewrites_user_int
            \seq_gput_left:NV \g__morewrites_write_seq \l__morewrites_tstr_tl
          }
      }
      {
        \prop_gpop:NVNT \g__morewrites_write_file_prop \l__morewrites_user_int \l__morewrites_
          { \__morewrites_closeout_now:nn { \l__morewrites_user_int } { \l__morewrites_interna
```

```
      }
  }
\cs_new_protected:Npn \__morewrites_closeout_now:nn #1#2
  {
    \__morewrites_tex_immediate:w \__morewrites_tex_openout:w \g__morewrites_iow = #2 \scan_st
    \group_begin:
      \int_set:Nn \tex_newlinechar:D { -1 }
      \tl_use:c { \__morewrites_user_tl_name:n {#1} }
      \tl_gclear:c { \__morewrites_user_tl_name:n {#1} }
    \group_end:
    \__morewrites_tex_immediate:w \__morewrites_tex_closeout:w \g__morewrites_iow
  }
```

(*End definition for* `\__morewrites_closeout:w`*,* `\__morewrites_closeout_now:`*, and* `\__morewrites_closeout_now:nn`*.*)

`\__morewrites_openout:w` `\__morewrites_openout_now:n`

In the immediate case find a file name, then allocate a TEX stream if possible, and otherwise point the user stream to a token list. In all cases, close the stream to avoid losing any material that TEX would have written, and empty the file by opening and closing it (actually that's done automatically by the primitive).

```
\cs_new_protected:Npn \__morewrites_openout:w
  {
    \s__morewrites
    \use_i:nn
      { \__morewrites_get_user:n { \__morewrites_openout_later:w } }
      { \__morewrites_get_user:n { \__morewrites_equals_file:N \__morewrites_openout_now:n } }
  }
\cs_new_protected:Npn \__morewrites_openout_now:n #1
  {
    \__morewrites_closeout_now:
    \seq_pop:NNTF \g__morewrites_write_seq \l__morewrites_tstr_tl
      {
        \prop_gput:NVV \g__morewrites_write_prop \l__morewrites_user_int \l__morewrites_tstr_t
        \__morewrites_tex_immediate:w \__morewrites_tex_openout:w \l__morewrites_tstr_tl \exp_
          = \tl_to_str:n {#1} \scan_stop:
      }
      {
        \__morewrites_empty_file:n {#1}
        \prop_gput:NVx \g__morewrites_write_file_prop \l__morewrites_user_int
          { \tl_to_str:n {#1} }
        \tl_gclear_new:c { \__morewrites_user_tl_name:n { \l__morewrites_user_int } }
      }
  }
```

(*End definition for* `\__morewrites_openout:w` *and* `\__morewrites_openout_now:n`*.*)

`\__morewrites_write:w` `\__morewrites_write_now:w` `\__morewrites_write_now:n`

In the immediate case we use `\__morewrites_write_now_open:n` if the stream points to a token list, and otherwise use the primitive, with the dummy stream 16 if closed (the text is then written to the terminal).

```
\cs_new_protected:Npn \__morewrites_write:w
  {
    \s__morewrites
    \use_i:nn
      { \__morewrites_get_user:n { \__morewrites_write_later:w } }
```

```
      { \__morewrites_get_user:n { \__morewrites_write_now:w } }
  }
\cs_new_protected:Npn \__morewrites_write_now:w
  {
    \__morewrites_user_to_tstr:NTF \g__morewrites_write_prop
      { \__morewrites_tex_immediate:w \__morewrites_tex_write:w \l__morewrites_tstr_tl \exp_st
      { \primargs_get_general_text:N \__morewrites_write_now:n }
  }
\cs_new_protected:Npn \__morewrites_write_now:n
  {
    \prop_get:NVNTF \g__morewrites_write_file_prop \l__morewrites_user_int \l__morewrites_inte
      { \__morewrites_write_now_open:n }
      { \__morewrites_tex_immediate:w \__morewrites_tex_write:w 16 }
  }
```

(*End definition for* `\__morewrites_write:w`*,* `\__morewrites_write_now:w`*, and* `\__morewrites_write_now:n`*.*)

`\__morewrites_write_now_open:n`
`\__morewrites_write_now_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 (after making sure we are allowed to alter it), 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 a token list whose name is constructed using `\__morewrites_user_tl_name:n`. 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_write_now_open:n #1
  {
    \bool_if:NF \g__morewrites_tmp_file_bool { \__morewrites_chk_file: }
    \__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:Nn \tex_endlinechar:D { -1 }
      \tex_openin:D \g__morewrites_ior = \g__morewrites_tmp_file_tl \scan_stop:
      \__morewrites_write_now_loop:
      \tex_closein:D \g__morewrites_ior
      \__morewrites_collect_gput_right:c
        { \__morewrites_user_tl_name:n { \l__morewrites_user_int } }
    \group_end:
    \__morewrites_empty_file:n { \g__morewrites_tmp_file_tl }
  }
\cs_new_protected:Npn \__morewrites_write_now_loop:
  {
    \etex_readline:D \g__morewrites_ior to \l__morewrites_internal_tl
    \ior_if_eof:NF \g__morewrites_ior
      {
        \__morewrites_collect:x
          {
            \__morewrites_tex_immediate:w \__morewrites_tex_write:w
              \g__morewrites_iow { \l__morewrites_internal_tl }
```

```
          }
        \__morewrites_write_now_loop:
      }
  }
```

(*End definition for* `\__morewrites_write_now_open:n` *and* `\__morewrites_write_now_loop:`.)

2.4.3 Delayed actions

`\__morewrites_later:n` `\__morewrites_later_do:n`

Store the action to be done at shipout in a token list, and non-immediately write the label `\g__morewrites_later_int` of the output operation to the temporary file.

```
\cs_new_protected:Npn \__morewrites_later:n #1
  {
    \int_gincr:N \g__morewrites_later_int
    \tl_const:cx
      {
        c__morewrites_later_
        \int_use:N \g__morewrites_later_int
        _tl
      }
      {
        \int_set:Nn \exp_not:N \l__morewrites_user_int
          { \exp_not:V \l__morewrites_user_int }
        \exp_not:n {#1}
      }
    \exp_args:NNx \__morewrites_tex_write:w \g__morewrites_iow
      { `( \int_use:N \g__morewrites_later_int ) }
  }
\cs_new_protected:Npn \__morewrites_later_do:n #1
  { \tl_use:c { c__morewrites_later_ \int_eval:n {#1} _tl } }
```

(*End definition for* `\__morewrites_later:n` *and* `\__morewrites_later_do:n`.)

`\__morewrites_closeout_later:`

If the user stream is a TEX stream, use the primitive, otherwise save `\__morewrites_closeout_now:` for later.

```
\cs_new_protected:Npn \__morewrites_closeout_later:
  {
    \int_compare:nNnTF \l__morewrites_user_int < { 19 }
      { \__morewrites_tex_closeout:w \l__morewrites_user_int }
      { \__morewrites_later:n { \__morewrites_closeout_now: } }
  }
```

(*End definition for* `\__morewrites_closeout_later:`.)

`\__morewrites_openout_later:w` `\__morewrites_openout_later:n`

If the user stream is a TEX stream use the primitive, otherwise find a file name and call `\__morewrites_openout_now:n` later.

```
\cs_new_protected:Npn \__morewrites_openout_later:w
  {
    \int_compare:nNnTF \l__morewrites_user_int < { 19 }
      { \__morewrites_tex_openout:w \l__morewrites_user_int }
      { \__morewrites_equals_file:N \__morewrites_openout_later:n }
  }
\cs_new_protected:Npn \__morewrites_openout_later:n #1
  { \__morewrites_later:n { \__morewrites_openout_now:n {#1} } }
```

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

`\__morewrites_write_later:w`
`\__morewrites_write_later:n`
`\__morewrites_write_later_aux:n`

For TEX streams use the primitive, otherwise find a general text and save it for later; the auxiliary is very similar to `\__morewrites_write_now:w`.

```
\cs_new_protected:Npn \__morewrites_write_later:w
  {
    \int_compare:nNnTF \l__morewrites_user_int < { 19 }
      { \__morewrites_tex_write:w \l__morewrites_user_int }
      { \primargs_get_general_text:N \__morewrites_write_later:n }
  }
\cs_new_protected:Npn \__morewrites_write_later:n #1
  { \__morewrites_later:n { \__morewrites_write_later_aux:n {#1} } }
\cs_new_protected:Npn \__morewrites_write_later_aux:n
  {
    \__morewrites_user_to_tstr:NTF \g__morewrites_write_prop
      { \__morewrites_tex_immediate:w \__morewrites_tex_write:w \l__morewrites_tstr_tl \exp_st
      { \__morewrites_write_now:n }
  }
```

(*End definition for* `\__morewrites_write_later:w`, `\__morewrites_write_later:n`, *and* `\__morewrites_write_later_aux:n`.)

2.4.4 Shipout business

In this section, we hook into the `\shipout` primitive, and redefine it to first build a box with the material to ship out, then perform

`\__morewrites_before_shipout:`
⟨*primitive shipout*⟩ ⟨*collected box*⟩
`\__morewrites_after_shipout:`

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.

`\__morewrites_before_shipout:`

Immediately before the shipout, we must open the writing stream `\g__morewrites_iow` (after making sure we are allowed to alter the auxiliary file).

```
\cs_new_protected:Npn \__morewrites_before_shipout:
  {
    \bool_if:NF \g__morewrites_tmp_file_bool { \__morewrites_chk_file: }
    \__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).

[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.

The looping auxiliary `\__morewrites_after_shipout_loop:ww` extracts the ⟨*operation*⟩ numbers from the file, and makes a token list out of those. This token list is then used 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: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 }
      \tl_set:Nx \l__morewrites_internal_tl
        {
          \exp_after:wN \__morewrites_after_shipout_loop:ww
          \tex_input:D \g__morewrites_tmp_file_tl \c_space_tl
        }
      \__morewrites_empty_file:n { \g__morewrites_tmp_file_tl }
      \exp_args:NNo
    \group_end:
    \tl_map_function:nN { \l__morewrites_internal_tl } \__morewrites_later_do:n
  }
\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:` *and* `\__morewrites_after_shipout_loop:ww`.)

`\__morewrites_shipout:w`
`\__morewrites_shipout_i:`
`\__morewrites_shipout_ii:`

Grab the shipped out box using `\setbox` and regain control using `\afterassignment`. There are two cases: either the box is given as `\box` or `\copy` followed by a number, in which case `\__morewrites_shipout_i:` is inserted afterwards at the same group level, or the box is given as `\hbox` (or `\vtop` and so on) and an additional `\aftergroup` is needed to reach a point where we can use the box saved in `\g__morewrites_shipout_box`.

```
\cs_new_protected: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: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:Npn \__morewrites_shipout_ii:
  {
    \__morewrites_before_shipout:
    \__morewrites_tex_shipout:w \tex_box:D \g__morewrites_shipout_box
    \__morewrites_after_shipout:
  }
```

(*End definition for* `\__morewrites_shipout:w`, `\__morewrites_shipout_i:`, *and* `\__morewrites_shipout_ii:`.)

`\shipout` `\__morewrites_tex_shipout:w`

The task is now to locate the shipout primitive, which may have been renamed and hooked into by many different packages loaded before morewrites. Any of those control sequences which are equal to the primitive are redefined to do `\__morewrites_shipout:w` instead. If the primitive is not located at all, the fallback is to hook into the control sequence `\shipout`.

```
\cs_gset_protected:Npn \__morewrites_tmp:w #1
  {
    \cs_if_exist:NF \__morewrites_tex_shipout:w
      { \cs_new_eq:NN \__morewrites_tex_shipout:w #1 }
    \cs_gset_eq:NN #1 \__morewrites_shipout:w
  }
\tl_map_inline:nn
  {
    \xyrealshipout@
    \org@shipout
    \PDFSYNCship@ut@ld
    \CROP@shipout
    \@soORI
    \tex_shipout:D
    \zwpl@Hship
    \o@shipout@TP
    \LL@shipout
    \Shipout
    \GXTorg@shipout
    \AtBegShi@OrgShipout
    \AtBeginShipoutOriginalShipout
    \shipout
  }
  {
    \str_if_eq_x:nnT
      { \cs_meaning:N #1 }
      { \token_to_str:N \shipout }
      { \__morewrites_tmp:w #1 }
  }
\cs_if_exist:NF \__morewrites_tex_shipout:w
  {
    \cs_new_eq:NN \__morewrites_tex_shipout:w \shipout
    \cs_gset_eq:NN \shipout \__morewrites_shipout:w
  }
```

(*End definition for* `\shipout` *and* `\__morewrites_tex_shipout:w`.)

2.4.5 Hook at the very end

`\__morewrites_close_all:`

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

```
\cs_new_protected:Npn \__morewrites_close_all:
  {
    \prop_map_function:NN \g__morewrites_write_prop
      \__morewrites_closeout_now:nn
    \prop_gclear:N \g__morewrites_write_prop
  }
```

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

`\__morewrites_close_all_at_end:nw` At the end of the run, we try very hard to put some material at the `\@@end`, just in case some other very late code writes to files that are not yet closed. This is tried at most 5 times, to avoid infinite loops in case two packages compete for that last place. The four `@` become two after l3docstrip.

```
\cs_new_protected:Npn \__morewrites_close_all_at_end:nw #1#2 \@@end
  {
    \int_compare:nNnTF {#1} > \c_zero
      { #2 \__morewrites_close_all_at_end:nw { #1 - 1 } }
      { \__morewrites_close_all: #2 }
    \@@end
  }
\AtEndDocument { \__morewrites_close_all_at_end:nw { 5 } }
```

(*End definition for* `\__morewrites_close_all_at_end:nw`.)

2.5 Redefining commands

2.5.1 Modified `\newwrite`

`\g__morewrites_alloc_read_int` `\g__morewrites_alloc_write_int` Counters to allocate user streams. Initialized to 18 so that the first user stream allocated by morewrites is 19. Indeed, 18 is reserved for shell commands and packages may expect 16 or 17 to write to the terminal.

```
\int_new:N \g__morewrites_alloc_read_int
\int_set:Nn \g__morewrites_alloc_read_int { 18 }
\int_new:N \g__morewrites_alloc_write_int
\int_set:Nn \g__morewrites_alloc_write_int { 18 }
```

(*End definition for* `\g__morewrites_alloc_read_int` *and* `\g__morewrites_alloc_write_int`.)

`\__morewrites_newread:N` Reimplementation of `\newread` but protected and using a counter `\g__morewrites_alloc_read_int` instead of what TEX/LATEX 2ε use.

```
\cs_new_protected:Npn \__morewrites_newread:N #1
  {
    \int_gincr:N \g__morewrites_alloc_read_int
    \int_set_eq:NN \allocationnumber \g__morewrites_alloc_read_int
    \cs_undefine:N #1
    \int_const:Nn #1 { \allocationnumber }
    \wlog
      {
        \token_to_str:N #1
        = \token_to_str:N \read \int_use:N \allocationnumber
      }
  }
```

(*End definition for* `\__morewrites_newread:N`.)

__morewrites_newwrite:N Same as for \newread.

```
\cs_new_protected:Npn \__morewrites_newwrite:N #1
  {
    \int_gincr:N \g__morewrites_alloc_write_int
    \int_set_eq:NN \allocationnumber \g__morewrites_alloc_write_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* __morewrites_newwrite:N.)

__morewrites_allocate:n Raise to #1 the number of \write streams allocated to morewrites.

```
\cs_new_protected:Npn \__morewrites_allocate:n #1
  {
    \prg_replicate:nn
      {
        \int_max:nn { 0 }
          {
            (#1) - \seq_count:N \g__morewrites_write_seq
            - \prop_count:N \g__morewrites_write_prop
          }
      }
      {
        \__morewrites_tex_newwrite:N \l__morewrites_tstr_token
        \seq_put_right:NV \g__morewrites_write_seq \l__morewrites_tstr_token
      }
  }
```

(*End definition for* __morewrites_allocate:n.)

2.6 User commands and keys

\morewritessetup Set whatever keys the user passes to \morewritessetup.

```
\cs_new_protected:Npn \morewritessetup #1
  { \keys_set:nn { __morewrites } {#1} }
```

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

file Because of our use of .initial:n, this code must appear after __morewrites_set_file:n is defined.

```
\keys_define:nn { __morewrites }
  {
    allocate .code:n = \__morewrites_allocate:n {#1} ,
    file .code:n = \__morewrites_set_file:n {#1} ,
    file .initial:n = \c_sys_jobname_str .mw
  }
```

(*End definition for* file. *This function is documented on page* ??.)

```
\openin
  \read
\readline
 \closein
 \newread
\immediate
 \openout
   \write
\closeout
\newwrite
```

```
\cs_gset_eq:NN \openin    \__morewrites_openin:w
\cs_gset_eq:NN \read      \__morewrites_read:w
\cs_gset_eq:NN \readline  \__morewrites_readline:w
\cs_gset_eq:NN \closein   \__morewrites_closein:w
\cs_gset_eq:NN \newread   \__morewrites_newread:N
\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* `\openin` *and others. These functions are documented on page* **??**.)

</package>