The termmenu package
Terminal-driven menu support*

Sean Allred†

Released 2015-05-25

This module provides simple support for terminal-driven menus in expl3.

Example of use

```
\termmenu_new:N \g_demo_termmenu
\termmenu_set_name:Nn \g_demo_termmenu { Demo }

\termmenu_add:Nnnn \g_demo_termmenu { d, duck }
  { Oh,~you~know...~:) }
  { \msg_term:n { Quack! } }

\termmenu_do:N \g_demo_termmenu

\bye
```

```
$ pdftex demo.tex
************************************************
* Demo
************************************************

The following commands are available:

    > d, duck
        Oh, you know... :)

\choice=d
************************************************
* Quack!
************************************************
```

*This file describes v0.2, last revised 2015-05-25.

†E-mail: tex@seanallred.com

Contents

Introduction

Menus are effectively documented property lists with a fancy, user-friendly version of `\prop_show:N`.

Part I
Interface Documentation

1 Creating and initializing menus

\termmenu_new:N

New: 2015-05-23

`\termmenu_new:N` ⟨*menu*⟩

Creates a new ⟨*menu*⟩ or raises an error if the name is already taken. The declaration is global. Initially, the menu will be empty.

\termmenu_set_name:Nn

New: 2015-05-23

`\termmenu_set_name:Nn` ⟨*menu*⟩ {⟨*name*⟩}

Give ⟨*menu*⟩ a human-friendly name. When a menu is being presented, ⟨*name*⟩ will appear as a title.

\termmenu_add:Nnnn

New: 2015-05-23

`\termmenu_add:Nnnn` ⟨*menu*⟩ {⟨*entry*⟩} {⟨*help text*⟩} {⟨*value*⟩}

Insert ⟨*entry*⟩ into a ⟨*menu*⟩ and provide ⟨*help text*⟩. When a menu is being presented, both ⟨*entry*⟩ and ⟨*help text*⟩ will be shown. When ⟨*entry*⟩ is being used, ⟨*value*⟩ will be associated with it.

To simplify the user experience, ⟨*entry*⟩ can be a comma-separated list of synonymous options. This can be used to create shortcuts to functionality: instead of always typing out `entry`, the end-user can simply say `o` if something like the following is used:

```
\termmenu_add:Nnnn \g_tmpa_termmenu { o, option } { ... } { ... }
```

2 Using menus

To allow for ad-hoc processing, menus are shown and acted upon in four separate phases: display, input, retrieval, insertion.

1. The display phase displays the menu to the end-user.
2. Then, an input routine is called upon to store information into a specific variable while prompting for a different one. (This is useful to keep the interface clean. The internal variable can be very strange-looking, but the user can see whatever you which them to see.)
3. The retrieval phase receives the user's input and looks it up in the list of option synonyms, returning the matching menu entry.
4. When the entry has been found, the appropriate value is either returned to the programmer or inserted into the input stream.

\termmenu_do:NNNNN
\termmenu_do:NNNN
\termmenu_do:NNN
\termmenu_do:NN
\termmenu_do:N

New: 2015-05-24

\termmenu_do:NNNNN ⟨*menu*⟩ ⟨*prompt token*⟩ ⟨*input tl*⟩ ⟨*entry tl*⟩ ⟨*value tl*⟩
\termmenu_do:NNNN ⟨*menu*⟩ ⟨*prompt token*⟩ ⟨*input tl*⟩ ⟨*entry tl*⟩
\termmenu_do:NNN ⟨*menu*⟩ ⟨*input tl*⟩ ⟨*value tl*⟩
\termmenu_do:NN ⟨*menu*⟩ ⟨*prompt token*⟩
\termmenu_do:N ⟨*menu*⟩

The `\termmenu_do:` family of functions are convenience wrappers around the basic display, prompt, and lookup functions.

`do:NNNNN` uses ⟨*menu*⟩ to prompt the user for ⟨*prompt token*⟩, setting their input to ⟨*input tl*⟩, setting the matched entry to ⟨*entry tl*⟩, and storing the associated value in ⟨*value tl*⟩.

`do:NNNN` is the same as `do:NNNNN`, but instead of storing the output in a token list, it is inserted into the input stread.

`do:NNN` is similar to `do:NNNNN`, but does not allow the programmer to set what the user sees as a prompt. Instead, the generic `\choice` name is used for ⟨*input tl*⟩. Additionally, the associated menu entry is not stored.

`do:NN` is like `do:NNNN`, but does not store either the user's input or its associated menu entry. The only parameter this takes is the ⟨*prompt token*⟩.

`do:N` is the simplest way to use a menu. It uses `\choice` for the ⟨*prompt token*⟩ and places the associated value in the input stream.

3 Input and output

\termmenu_prompt:NN
\termmenu_prompt:N

New: 2015-05-23
Updated: 2015-05-24

\termmenu_prompt:NN ⟨*input tl*⟩ ⟨*prompt token*⟩
\termmenu_prompt:N ⟨*input tl*⟩

Read a value for ⟨*input tl*⟩ showing ⟨*prompt token*⟩ to the user. When ⟨*prompt token*⟩ isn't provided, it defaults to `\choice`.

\l_termmenu_prompt_tl

New: 2015-05-23

This is the message displayed to the user when a menu is presented. The default value is `The~following~commands~are~available:`.

4 Inspection

\termmenu_show:N

New: 2015-05-23

\termmenu_show:N ⟨*menu*⟩

Show the contents of ⟨*menu*⟩. Right now, this is the same as `\prop_show:N`.

\termmenu_get_name:NN

New: 2015-05-23
Updated: 2015-05-24

\termmenu_get_name:NN ⟨*menu*⟩ ⟨*name tl*⟩

Retrieve the name of ⟨*menu*⟩ and place it in ⟨*name tl*⟩.

\termmenu_entry:NnN
\termmenu_entry:NVN
New: 2015-05-25

\termmenu_entry:NnN ⟨*menu*⟩ {⟨*option*⟩} ⟨*entry tl*⟩

Finds the entry that ⟨*option*⟩ would refer to in ⟨*menu*⟩ and places it in ⟨*entry tl*⟩.

\termmenu_doc:NnN
New: 2015-05-25

\termmenu_doc:NnN ⟨*menu*⟩ {⟨*entry*⟩} ⟨*doc tl*⟩

Finds the documentation for ⟨*entry*⟩ in ⟨*menu*⟩ and places it in ⟨*doc tl*⟩.

\termmenu_value:NnN
\termmenu_value:NVN
New: 2015-05-25

\termmenu_value:NnN ⟨*menu*⟩ {⟨*entry*⟩} ⟨*value tl*⟩

Places ⟨*menu*⟩'s value for ⟨*entry*⟩ in ⟨*value tl*⟩.

5 Internal variables and functions

\g__termmenu_names_prop
New: 2015-05-23

This property list stores the names of each menu. The keys of the property lists are menus (e.g., \g_demo_termmenu) and the values are their names.

\l__termmenu_spec_tl
New: 2015-05-23

Generally speaking, this scratch variable stores lower-level information about a menu (or one of its entries).

\g__termmenu_doc_tl
New: 2015-05-23

This scratch variable stores the documentation for an entry. It is necessary because none of f, x, o-type expansions seem to work where they need to. Patches/pull requests welcome.

\l__termmenu_value_tl
New: 2015-05-24

This scratch variable stores the value of an entry. It is immediately inserted back into the input stream.

\l__termmenu_tmp_tl
New: 2015-05-24

This scratch variable holds values returned by the various do: convenience macros. This value serves as a sort of 'trash bin' to throw away some unwanted values.

\g__termmenu_tmp_tl
New: 2015-05-24

This scratch variable is used to help set ⟨*tl var*⟩ from \termmenu_find:NnN.

\g__termmenu_opt_bool
New: 2015-05-24

This variable is used to signal if the user's input was able to match against a known option.

\termmenu_display:N
New: 2015-05-24

\termmenu_display:N ⟨*menu*⟩

Writes out ⟨*menu*⟩ to the terminal. No associated actions are performed. If ⟨*menu*⟩ has a name (i.e., an entry in __termmenu_names_prop), print it as well.

Part II
Implementation

Before I begin, I want to establish a few definitions:

menu The over-arching data structure that holds (nearly) all related information for a given menu.

entry A full, comma-separated specification of valid inputs for a given menu item.

option A valid input for an entry.

value The associated action for an entry.

prompt token The token read in by TEX on a low level; this token's name is presented to the user in output.

1 Creation and initialization

```
1 ⟨*package⟩
2 ⟨@@=termmenu⟩
```

\termmenu_new:N
\termmenu_show:N
\termmenu_add:Nnnn

Each menu is implemented as a property list of options mapped to documentation and actions. Each option is a property list key. Since the there is no good way to distinguish between when documentation ends and an associated action begins, the documentation is placed in a group at the head of the key's value.[1]

```
3 \cs_set_eq:NN \termmenu_new:N \prop_new:N
4 \cs_set_eq:NN \termmenu_show:N \prop_show:N
5 \cs_new_nopar:Nn \termmenu_add:Nnnn
6   { \prop_put:Nnn #1 {#2} { {#3} #4 } }
```

(*End definition for* `\termmenu_new:N`, `\termmenu_show:N`, *and* `\termmenu_add:Nnnn`. *These functions are documented on page 3.*)

2 Convenience functions

\l__termmenu_tmp_tl
\l__termmenu_value_tl

These scratch variables are used in the following code. You may be reviewing the code and thing to yourself, "Wait, couldn't you use just one variable?" Rest assured that you cannot: `\termmenu_do:NNNN` uses `\l__termmenu_value_tl` to insert the value code into the input stream. Using the same variable for this would be at best confusing/ unmaintainble. Besides, `\l__termmenu_tmp_tl` is supposed to be a variable for unwanted values.

```
7 \tl_new:N \l__termmenu_tmp_tl
8 \tl_new:N \l__termmenu_value_tl
```

[1] This could also be done with sequences, but I consider that unnecessary overhead and complication.

(*End definition for* `\l__termmenu_tmp_tl` *and* `\l__termmenu_value_tl`*. These variables are documented on page 5.*)

`\termmenu_do:NNNNN` `\termmenu_do:NNNN` `\termmenu_do:NNN` `\termmenu_do:NN` `\termmenu_do:N`

These functions make menus easier to use by choosing some sane defaults and running through the entire flow. Each of these macros is ultimately based on the most general one, `\termmenu_do:NNNNN`. This results in some waste of work, but those extra gets/sets pale in comparison to wrapping the menu output, executing whatever the menu entry stands for, *actually waiting for the user to make a choice*, etc.

```
\cs_new_nopar:Nn \termmenu_do:NNNNN
  {
    \termmenu_display:N #1
    \termmenu_prompt:NN #2 #3
    \termmenu_entry:NVN #1 #2 #4
    \termmenu_value:NVN #1 #4 #5
  }
\cs_new_nopar:Nn \termmenu_do:NNNN
  {
    \tl_clear:N \l__termmenu_value_tl
    \termmenu_do:NNNNN #1 #2 #3 #4 \l__termmenu_value_tl
    \tl_use:N \l__termmenu_value_tl
  }
\cs_new_nopar:Nn \termmenu_do:NNN
  { \termmenu_do:NNNNN #1 #2 \choice \l__termmenu_tmp_tl #3 }
\cs_new_nopar:Nn \termmenu_do:NN
  { \termmenu_do:NNNN #1 \l__termmenu_tmp_tl #2 \l__termmenu_tmp_tl }
\cs_new_nopar:Nn \termmenu_do:N
  { \termmenu_do:NN #1 \choice }
```

(*End definition for* `\termmenu_do:NNNNN` *and others. These functions are documented on page 4.*)

3 Retrieving values

`\l__termmenu_spec_tl`

`\l__termmenu_spec_tl` is used when retrieving data from property lists. It's used quite a bit in the following functions.

```
\tl_new:N \l__termmenu_tmp_tl
\tl_new:N \l__termmenu_value_tl
\tl_new:N \l__termmenu_spec_tl
```

(*End definition for* `\l__termmenu_spec_tl`*. This variable is documented on page 5.*)

`\termmenu_value:NnN` `\termmenu_value:NVN`

This function retrieves the value for entry `#2` in menu `#1` and places it in `#3`. Remember that the documentation and value are stored together in the property list. Since the documentation is kept in a group, we can use `\tl_tail:N` to grab just the desired value.

```
\cs_new_nopar:Nn \termmenu_value:NnN
  {
    \prop_get:NnN #1 {#2} \l__termmenu_spec_tl
    \tl_set:Nf #3 { \tl_tail:N \l__termmenu_spec_tl } %@todo test expansion
  }
\cs_generate_variant:Nn \termmenu_value:NnN { NVN }
```

(*End definition for* `\termmenu_value:NnN` *and* `\termmenu_value:NVN`. *These functions are documented on page 5.*)

4 Entry documentation

`\g__termmenu_doc_tl` This variable stores the help text for an option.

```
\tl_new:N \l__termmenu_doc_tl
```

(*End definition for* `\g__termmenu_doc_tl`. *This variable is documented on page 5.*)

`\termmenu_doc:NnN` These functions retrieve the help text for entry `#2` in `#1` and place it in `#3`. Keep in mind the structure of `\l__termmenu_names_prop`.

```
\cs_new_nopar:Nn \termmenu_doc:NnN
  {
    \prop_get:NnN #1 {#2} \l__termmenu_spec_tl
    \tl_set:Nf #3 { \tl_head:N \l__termmenu_spec_tl } %@todo test expansion
  }
```

(*End definition for* `\termmenu_doc:NnN`. *This function is documented on page 5.*)

5 Entry lookup

`\g__termmenu_tmp_tl` This scratch variable is used to escape the groups of mapping constructs like `\prop_map_inline:Nn`.

```
\tl_new:N \g__termmenu_tmp_tl
```

(*End definition for* `\g__termmenu_tmp_tl`. *This variable is documented on page 5.*)

`\g__termmenu_opt_bool` This scratch variable is used to signal if a suitable option was found when searching for a match based on user input. Since it is used inside two inlined mappings, it is most straightforward for all assignments to be global.

```
\bool_new:N \g__termmenu_opt_bool
```

(*End definition for* `\g__termmenu_opt_bool`. *This variable is documented on page 5.*)

`\termmenu_entry:NnN`
`\termmenu_entry:NVN` Using the menu in `#1`, find a match for `#2` and stick that match in `#3`.

```
\cs_new_nopar:Nn \termmenu_entry:NnN
  {
    \prop_map_inline:Nn #1
      {
```

If the input value is in the list of synonyms, set our output value to the full list, indicate that we've found a match, and break out of the loop. Note that we use global variables to free ourselves from the confines of the group.

```
        \clist_if_in:nnT {##1} {#2}
          {
            \tl_gset:Nn \g__termmenu_tmp_tl {##1}
            \bool_gset_true:N \g__termmenu_opt_bool
```

```
            \prop_map_break:
          }
      }
```

If we haven't set `\g__termmenu_opt_bool` by the time we've finished looking, we never found anything. Set `#3` to `\q_no_value` to indicate the lack of a match and reset the value of `\g__termmenu_opt_bool`.

```
    \bool_if:NTF \g__termmenu_opt_bool
      { \tl_set_eq:NN #3 \g__termmenu_tmp_tl }
      { \tl_set:Nn #3 { \q_no_value } }
    \bool_gset_false:N \g__termmenu_opt_bool
  }
\cs_generate_variant:Nn \termmenu_entry:NnN { NVN }
```

(*End definition for* `\termmenu_entry:NnN` *and* `\termmenu_entry:NVN`. *These functions are documented on page* 5.)

6 Retrieving input

`\l_termmenu_prompt_tl`

This public variable contains the text to be used as a prompt when displaying a menu. It is given a reasonable default.

```
\tl_new:N \l_termmenu_prompt_tl
\tl_set:Nn \l_termmenu_prompt_tl
 { The~following~commands~are~available: }
```

(*End definition for* `\l_termmenu_prompt_tl`. *This variable is documented on page* 4.)

`\termmenu_prompt:NN`
`\termmenu_prompt:N`

This function is a little interesting. In order to keep the end-user's interface clean, we can't simply prompt for the destination variable. Say, if the destination variable were something like `\l__some_confusing_variable_name`, this would cause TEX to display that confusing variable name to the end-user as part of the prompt. Instead, we ask for `#2` (starting a new group to avoid clobbering any existing definition) and TEX puts the user's input in `#2`. Now we have to get this value outside the group and into `#1`. The `\expandafter\endgroup` trick works nicely here (and is the official[2] way to do this).

```
\cs_new_nopar:Nn \termmenu_prompt:NN
  {
    \group_begin:
      \termmenu_prompt:N #2
    \exp_args:NNNV \group_end:
    \tl_set:Nn #1 #2
  }
\cs_new_nopar:Nn \termmenu_prompt:N
  { \ior_get_str:NN \c_term_ior #1 }
```

(*End definition for* `\termmenu_prompt:NN` *and* `\termmenu_prompt:N`. *These functions are documented on page* 4.)

[2] http://tex.stackexchange.com/a/246542

7 Names

\g__termmenu_names_prop Globally define the property list to store menu names.

```
\prop_new:N \g__termmenu_names_prop
```

(*End definition for* `\g__termmenu_names_prop`. *This variable is documented on page* 5.)

\termmenu_get_name:NN

```
\cs_new_nopar:Nn \termmenu_get_name:NN
 { \prop_get:NnN \g__termmenu_names_prop {#1} #2 }
```

(*End definition for* `\termmenu_get_name:NN`. *This function is documented on page* 4.)

\termmenu_set_name:Nn This function names a menu. An entry is placed in `\g__termmenu_names_prop` with the menu `#1` as a key and its name `#2` as the value.

```
\cs_new_nopar:Nn \termmenu_set_name:Nn
 { \prop_put:Nnn \g__termmenu_names_prop {#1} {#2} }
```

(*End definition for* `\termmenu_set_name:Nn`. *This function is documented on page* 3.)

8 Display

\termmenu_display:N Perhaps the only truly complicated part of this package is this function. Let's generate some new terminal output variants to make our lives easier.

```
\cs_generate_variant:Nn \iow_term:n { V }
\cs_generate_variant:Nn \msg_term:n { V }
\cs_new_nopar:Nn \termmenu_display:N
  {
```

First, we retrieve the name of the menu. If it does not exist, print a generic "Menu" header. If it does exist, use the menu's title as the header.

```
    \termmenu_get_name:NN #1 \l__termmenu_spec_tl
    \quark_if_no_value:NTF \l__termmenu_spec_tl
      { \msg_term:n { Menu } }
      { \msg_term:V \l__termmenu_spec_tl }
```

Display the prompt. Note that `\iow_term:n` without an argument will simply output one blank line.

```
    \iow_term:n { }
    \iow_term:V \l_termmenu_prompt_tl
    \iow_term:n { }
```

We want to display each option with its help text in a way that is easy to read.

```
    \prop_map_inline:Nn #1
      {
```

For each entry `##1` in the menu `#1`, send wrapped output to the terminal that contains the entry `\l__termmenu_tmp_clist` indented by four spaces, a new line, and the documentation `\l__termmenu_tmp_tl`, all wrapped with a running indent of eight spaces. Note that since we introduce a new line with the `\\` macro in `\iow_wrap:nnnN`, we get the first indentation of the documentation for free.

```
        \clist_set:Nn \l__termmenu_tmp_clist {##1}
        \termmenu_doc:NnN #1 {##1} \l__termmenu_tmp_tl
        \iow_wrap:nnnN
          {
            \prg_replicate:nn {4} { \iow_char:N \  }
            > ~ \clist_use:Nn \l__termmenu_tmp_clist { ,~ } \\
            \tl_use:N \l__termmenu_tmp_tl
          } { \prg_replicate:nn {8} { \  } } { } \iow_term:n
      }
  }
```

(*End definition for* `\termmenu_display:N`. *This function is documented on page* 5.)

⟨/package⟩