The joy of \csname...\endcsname

Amy Hendrickson

Abstract

A surprisingly useful tool, \csname–\endcsname, offers many opportunities for interesting and useful macros, especially when it is convenient to dynamically generate a series of definitions.

Trivially a series of \csname definitions may be used to produce endnotes, but there are more interesting and complex constructions as well.

Another example shows how \csname may be used for on-line report generation. In this instance, we dynamically generate hyperlinked tabs for custom risk analyses of particular stocks chosen on-line by the client. We can use these named tabs to build a hyperlinked Table of Contents on the fly.

A similar process may be used to produce hyperlinked, tabbed, documentation.

The final example shows how definitions made with \csname can be used to send a set of definitions to an auxiliary file, where each new definition contains the current page number in its name, and a number as its definition.

This allows the dynamic redefinition of the command for a particular page, *within the auxiliary file*, depending on whether the value of the new definition is higher than the value of the previous definition for the same page.

Code will be shown for each of these methods to dynamically generate macros using \csname.

1 The basics

\csname and \endcsname are TEX primitives that allow us to define and call macros. If we compare a definition made with and without \csname we can see that initially a definition made with \csname is not much different than one made with \def. For instance, if we compare these two definitions,

1. **A definition with** \def:
   \def\puppy{Toy Poodle}
   and the new macro is called by writing: \puppy,
2. **A definition with** \csname...\endcsname:
   \expandafter\def\csname puppy\endcsname
   {Toy Poodle}
   called with: \csname puppy\endcsname

we find the results in either case to be 'Toy Poodle'. So, why bother with \csname...\endcsname?

2 Useful characteristics

As we will see, \csname has several of characteristics making it uniquely useful:

1. We can use \csname to find out if a command has been defined, since an undefined command is equal to \relax.

   As an example, we can make a conditional that tests to see if a command has been defined and make choices based on the result:

```
\expandafter\ifx
\csname anycommand \endcsname\relax
<do this>\else<do that>\fi
```

2. \csname allows us to define and call commands that may be composed of numbers, symbols, and other commands, unlike the basic command for making definitions, \def, which must use only letters for the name of a new definition.

   For example, this is a valid definition that uses symbols and a number in its name:

```
\expandafter\def\csname $&3\endcsname{Hi!}
```

   Commands made with non-letters must be called using \csname–\endcsname. In this case, \csname $&3\endcsname must be used, and produces: 'Hi!'.

   Another example comes from latex.ltx, the basic LaTeX macro set, a definition that uses \csname for making macros that might have characters other than letters in their name:

```
\def\@namedef#1{\expandafter\def\csname
                #1\endcsname}
```

   \@namedef{} is used widely in LaTeX code. As one example, \@namedef{} is used in the macro for making labels for cross-referencing. This is why you can make a label that looks like this, \label{fig1}, where the argument includes a number.

3. A macro argument may be used within \csname. As an example, again from latex.ltx:

```
\def\setcounter#1#2{\@ifundefined{c@#1}%
  {\@nocounterr{#1}}%
  {\global\csname c@#1\endcsname#2\relax}}
```

   used, e.g.: \setcounter{page}{201}

   The macro checks to see if there is a counter called \c@#1. If there is no counter with that name it will give an error message; if there is a defined counter, it uses \csname to call the counter and sets it to the number given as the second argument. In this example, it sees that a counter named \c@page exists, so sets it equal to the second argument, '201' in this example.

   Generic macro. The example above shows \csname...\endcsname being used to make a kind of generic macro since it will have the flexibility to be used with any previously defined LaTeX counter.

4. **Expand commands within \csname.**
   Here's where things get interesting. We can expand commands within a definition name made with \csname...\endcsname. This opens up many complex possibilities. For one set of possibilities, we can include a counter in the name of a new definition.

In this article we'll explore a number of ways in which we can use \csname...\endcsname with counters.

2.1 Dynamic macro building

We can use a counter within \csname...\endcsname to make a series of macros, a new macro every time the counter is advanced.

We do this by including a definition, made using \csname...\endcsname with a counter in its name, within the body of another definition. The outer definition advances a counter every time it is used, producing a new and unique inner macro every time it is called.

For example, we can make a command that will make more commands in this way:

```
\newcount\applenum
\def\applename#1{\global\advance\applenum by 1
  \expandafter\def\csname apple\the\applenum
  \endcsname{**#1**}}
```

Every time we use the \applename{} macro, we define a new macro, named \apple1, \apple2 and so on.

Using a loop to call the macros

To access the newly made inner macro we can use a loop, which advances a counter in each iteration, and calls the inner macro using the current state of the counter as part of the macro name.

To call the macros made with the \applename macro above, we test to see if \applename⟨*number*⟩ is defined. If defined, we call the command using the current state of the \loopnum counter in the body of the name of the command; else, end the loop.

```
\newcount\loopnum
\loopnum=1
\loop\expandafter\ifx
\csname apple\the\loopnum\endcsname\relax
\else
  \csname apple\the\loopnum\endcsname\
  \global\advance\loopnum by 1
\repeat
```

Used:

```
\applename{Macintosh}\applename{Gala}
```

Results:

Macintosh **Gala**

3 Endnotes example

For our first real world example we will use this tool to make endnotes. In this example we want to change the definition of \footnote so that it produces endnotes rather than footnotes. We do this by making an endnote definition that makes a new definition every time it is called.

We start with a new counter to be used by our endnotes, \endnum. In the \endnote macro we advance the \endnum counter, then raise and print the number in the text for our endnote number.

Next we make a construction with \csname that builds a new definition, using the current state of the \endnum counter. This new definition will be used to save the text of the endnote.

```
\newcount\endnum
\def\endnote#1{\global\advance\endnum by 1
  $^{\the\endnum}$%
%%
%% Here we make the new definition using
%% \the\endnum in the definition name so that
%% each new definition is unique:
%%
\long\expandafter
\def\csname endnote\the\endnum\endcsname{%
  \small\leftskip=12pt\relax\parindent=-12pt
  \indent\hbox to 12pt{\the\loopnum.\hfill}%
  %%
  %% Here we save the text of the endnote:
  #1%
  \strut\vskip2pt}}
```

Now we set \footnote to be equal to \endnote, so every time \footnote is used, the command actually called is \endnote:

```
\let\footnote\endnote
```

To print the endnotes, we make a loop that advances a counter with every iteration. That counter is used within the name of the definition made with \csname...\endcsname. The loop continues until it comes to an undefined endnote, thus cycling through every defined endnote. An example is shown in figure 1.

```
\newcount\loopnum
\def\printendnotes{\global\loopnum=1
%%
%% Test to see if any end notes have been
%% defined; If so, provide the title and
%% start loop; if not, do nothing.
%%
\expandafter\ifx
\csname endnote\the\loopnum\endcsname\relax
            \else
\subsection*{Endnotes}\everypar{}
\vskip6pt
\small\leftskip=12pt
```

```
``A day of dappled seaborne clouds.%
\footnote{Quotation from James Joyce's
`Portrait of the Artist as a Young Man'.}
The phrase and the day and the scene
harmonised in a chord. Words. Was it
their colours? He allowed them to glow
and fade, hue after hue: sunrise gold, the
russet and green of apple orchards, azure
of waves, the greyfringed fleece of
clouds.\footnote{The Bloomsday
celebration in Dublin this year features a
concert of compositions honoring Joyce.}

\printendnotes
```

"A day of dappled seaborne clouds.[1] The phrase and the day and the scene harmonised in a chord. Words. Was it their colours? He allowed them to glow and fade, hue after hue: sunrise gold, the russet and green of apple orchards, azure of waves, the greyfringed fleece of clouds.[2]

Endnotes

1. Quotation from James Joyce's 'Portrait of the Artist as a Young Man'.
2. The Bloomsday celebration in Dublin this year features a concert of compositions honoring Joyce.

Figure 1: Testing the endnote commands

```
%% Loop continues until it finds an
%% undefined endnote
%%
\loop\expandafter\ifx
\csname endnote\the\loopnum\endcsname\relax
                  \else
%% Print endnote
\csname endnote\the\loopnum\endcsname
\vskip2pt
%%
%% Reset: redefine current endnote to \relax
%% preventing this definition from being
%% used the next time \printendnotes is called.
%%
\global\expandafter
\let\csname endnote\the\loopnum\endcsname\relax
%%
\global\advance\loopnum by 1
\repeat
\fi
%% \fi ends test at beginning of this macro
%% to see if any endnotes have been defined.
}
```

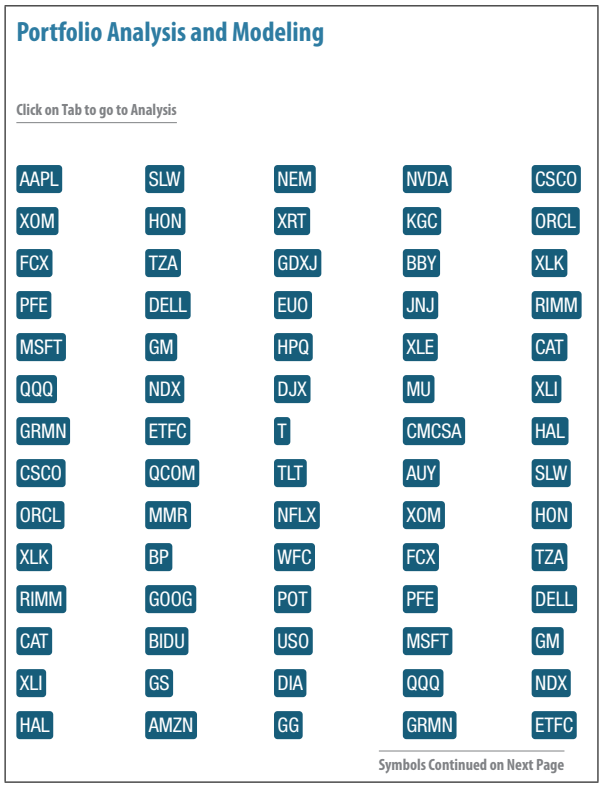

Figure 2: One form of automated online report generation; this is a draft version of customized financial reporting. Each symbol is automatically generated and is hyperlinked to the appropriate page of the report. The company analyzed depends on input from the client; the symbols and their linking is done through macros utilizing `\csname`.

4 Example: On-line report generation

A somewhat similar construction may be used to make hyperlinked tabs for on-line report generation (figure 2). (The actual reports use color, too.)

This set of macros is used to automate the naming of hypertargets so that we can hyperlink to them on the first page of the report, using a `\csname` construction and a loop, and using Ti*k*Z for making the hyperlinked tab.

The name and number of companies analyzed is determined by the client who submits a request online. Each company's analysis will start on a titled new page. Part of the definition for the title of a report will be the command `\maketab{#1}`.

`\maketab` takes a stock symbol as its argument, and generates a hypertarget so that we can link to it from the beginning of the report, in the equivalent of the table of contents page, using the same `\codenum` counter. Then it makes a new definition with `\csname` and the `\codenum` counter in its name,

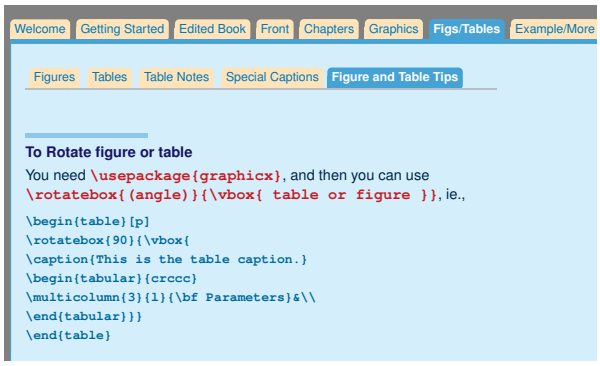

Figure 3: A similar technique is used for making hyperlinked tabbed documentation, where the hypertargets are made with `\csname` and a counter, accessed with hyperlinks named with `\csname` and a counter. The tabs in this example are also made with Ti*k*Z.

with the stock symbol as its definition, and sends it to the `.aux` file.

```
\def\maketab#1{\global\advance\codenum by 1
  \hypertarget{link\the\codenum}{}%
  \immediate\write\@auxout{\string\expandafter
    \string\gdef\string\csname\space
    tab\the\codenum\string\endcsname{#1}}}
```

Once we have this in place we can use our loop construction for the first, and possibly continuing, pages to build the hyperlinked tabs. `\gettabs` uses a loop to call the individual tabs, as long as there is one defined. This can continue over a number of pages if necessary.

```
\begin{multicols}{5}
\loopnum=1 \gettabs
\end{multicols}
```

As you can see, `\gettabs` is where the work is done. Here is how it is defined:

```
\def\gettabs{\loop
\expandafter\ifx
\csname tab\the\loopnum\endcsname\relax
            \else
\vskip6pt\hbox to 1in{%
%%
%% \hyperlink takes two arguments;
%% the first the name of the hypertarget,
%% and the second, the text that will link
%% to the hypertarget when clicked:
\hyperlink{link\the\loopnum}%
{\plaintab{\csname tab\the\loopnum\endcsname}%
 \hskip12pt}%
\hfill}%% <== end \hbox started above
\global\advance\loopnum by 1
            \repeat}
```

If you are interested in how to make the tab with Ti*k*Z, here is that code:

```
\definecolor{dkblue}{cmyk}{.9,.53, .32, .2}
\def\plaintab#1{%
\hbox{\normalsize\sf
\begin{tikzpicture}
[rounded corners=3pt, inner sep=3pt]%
   \node[rectangle,fill=dkblue]
{\Large\sf\color{white}
\vrule depth 3pt width 0pt height 15pt \relax
#1};
\end{tikzpicture}}}
```

A similar technique can be used to produce tabbed documentation, as shown in figure 3. For the full example, please see `http://www.texnology.com/docs.pdf`.

5 Another `\csname` technique, for classification levels

The problem:

When producing a classified document, the highest level of classification (secret, top secret, etc.) on any particular page must appear at the top of that page. When the classification level is given, the user doesn't know the page on which it will appear. In addition, the user doesn't know in advance whether this particular classification level is the highest on that page.

The solution:

We can use a `\write` to be sure that we know the page number where the markup has appeared, so the macro for making classification level markup will send the level along with the page number to an auxiliary file, using a `\write` command.

Now we have a page number and a level appearing on that page. However, we still don't know which level is highest for the particular page.

The second part of the solution involves defining the highest level for a particular page, *in the auxiliary file*, by also sending code for comparing levels on a particular page, and making a definition for the particular page *only if the present level is the highest for that page number*. When the auxiliary file is input, the next time LaTeX is run on the root file, the definition of the highest level on each page has been defined.

There are many more complications in the full problem. For instance, how do we pass information on the level of a paragraph that has broken over pages, so that the part of the paragraph on the second page will contribute to the calculation of the highest level on the second page? For the sake of brevity, we'll consider only the general mechanism here.

5.1 Setting up

We use a `\write` for every instance where a classification level is written in the text with the command `\secmark`. `\write` is only activated after the page is made up, so we are sure that we will be using the correct page number when we send the information to the auxiliary file. Since we will have many `\write` commands in the `.tex` document, we will write to a new auxiliary file, `\jobname.lev`, instead of using standard LaTeX auxiliary file, `\jobname.aux`. We name the new write:

```
\newwrite\collect
```

5.2 The counter to be used

The next item we need is a counter to use when defining our `\csname` commands. Since we want a command that has the current page number in its name, we would be tempted to use the LaTeX page counter, `\c@page`.

However, in the common case where the beginning of the document uses roman numerals, and the body of the document uses arabic numerals, we would have the unfortunate result of having multiple pages with the same page number.

So instead, we make a new counter, and call it `\superpage`:

```
\newcount\superpage
```

5.3 Using `\shipout`

`\shipout` is the TeX primitive that is called every time a page is completed. We use `\shipout` to generalize this solution, so that this system will work independently of any page style, and its headers and footers.

We can use `\shipout` to advance the counter called `\superpage`. This gives us a new number for every page, continuous through the document. Now `\shipout` can be used to print the classification term on the top and bottom of the page using `\superpage` as the counter found in the name that has been defined with `\csname`...`\endcsname`.

5.4 Doing the `\write`s

The `\secmark` macro works by sending a definition for the classification level on a particular page to `\jobname.lev` file, using a `\write` associating the page number with the level given. The `\write` will not be activated until the page is made up, so we are guaranteed to have the correct page number sent to the `\jobname.lev` file. This works as well for figure or table floats, since `\write` will send out the information to the `.lev` file only after the page is made up, and the page where the floats will appear has been determined.

The `\write` sends information to the auxiliary file, `\jobname.lev`, including several conditional tests. The command looks messy and verbose because when the write is made, we have to stop the expansion of many commands by preceding each one with `\string`, except for those few commands that we want to expand immediately; in this case, the super page number:

```
\write\collect{%% ^^J makes a blank line
%% in the \jobname.lev file so that
%% it is easier to see where each test ends:
^^J^^J
%%
\string\expandafter\string\ifx\string\csname%
\space LevelOnSuperPage\the\superpage
\string\endcsname\string\relax
\string\expandafter\string\gdef\string\csname
\space LevelOnSuperPage\the\superpage
\string\endcsname{#1}
\string\else
\string\ifnum \string\csname\space
LevelOnSuperPage\the\superpage\string\endcsname
\string< #1
\string\expandafter\string\gdef\string\csname
\space LevelOnSuperPage\the\superpage
\string\endcsname{#1}\string\fi\string\fi
^^J}%
```

... which might make more sense when we see how the code looks by the time it is expanded and appears in the `\jobname.lev` file. Here, the level sent for page 5 is '2'.

```
\expandafter\ifx
\csname LevelOnSuperPage5\endcsname\relax
\expandafter\gdef
\csname LevelOnSuperPage5\endcsname{2}
\else\ifnum\csname LevelOnSuperPage5\endcsname<
2 \expandafter\gdef
\csname LevelOnSuperPage5\endcsname{2}
\fi\fi
```

This process can be repeated as many times as needed for each page, with only the highest number, determined by each test, being used to define `\csname LevelOnSuperPage?\endcsname`.

Then, when `\jobname.lev` is brought into the base `.tex` file the next time LaTeX is run on the document, it will include a series of unique macros, one for each page in the document where a classification mark has been used, defining the highest number given for that page. Since the definition is made with `\csname`...`\endcsname` we can have the superpage number contained in the name of the definition. This allows us to call the definition using the current superpage number in the `\csname`...`\endcsname`, in the shipout.

5.5 Using the level information

We can use these definitions with every shipout, with the macro `\makeclassification` being called at the top and bottom of the page. Here is its definition:

```
\def\makeclassification{%
\vbox{\baselineskip=12pt
%% Is there a definition for this page?
\expandafter\ifx
\csname LevelOnSuperPage\the\superpage
\endcsname\relax
%% if not:
\centerline{}
\else
%% if there is a definition:
\centerline{%
\ChangeNumIntoClassification{%
\expandafter\csname
LevelOnSuperPage\the\superpage
\endcsname}}\vskip3pt\fi}}
```

`\ChangeNumIntoClassification`, seen above, uses the definition of

```
\csname LevelOnSuperPage\the\superpage
\endcsname
```

as its argument, which will yield a number from 1 to 4. This allows us to use `\ifcase` to trivially change that number into the classification term:

```
\def\ChangeNumIntoClassification#1{%
\ifcase#1\or Unclassified \or Classified
\or Secret \or Top Secret
\else ! Please Run LaTeX Again to Get the
Classification Level !
\fi}
```

And now we will have the highest classification level reliably appearing on top of each page.

Summary: Ways in which `\csname` is exceptionally useful

1. Testing to see if a macro has been defined.
2. Making a macro that has characters other than letters in its name, e.g., a cross referencing label.
3. Making a generic macro that can be modified with the argument of another macro.
4. Generating new macros by using a counter in the name made with `\csname`...`\endcsname`.
5. Calling macros made with `\csname` with a loop. The loop may be stopped by testing to see if the most recent `\csname` ⟨*counter*⟩`\endcsname` combination has been defined.

   Using this method to stop looping has the advantage that we don't need to know in advance how many definitions were made, and we will cycle through all available definitions before ending the loop.
6. A `\csname`...`\endcsname` definition including a counter in its name can be used to generate a series of hypertext targets automatically.
7. Definitions can be made using the page number as part of the name, which can be called by the output routine.
8. Finally, we have the technique of sending information to an auxiliary file with a `\write` and making new `\csname` ⟨*counter*⟩`\endcsname` definitions in the body of the auxiliary file, depending on the results of a conditional test. When the auxiliary file is input into the root `.tex` file, we can then use the resulting definition in a variety of ways.

`\csname` in the future

More than a coding oddity, `\csname`...`\endcsname` is a workhorse, allowing many constructions that wouldn't otherwise be available.

Likely there are many more opportunities to use these techniques, particularly with off-label uses for LaTeX such as report generation, or building e-documents on the fly, and other web-oriented macro writing projects.

Enjoy!

(The slides for the TUG 2012 conference talk are available at `http://www.texnology.com/talk.pdf`.)

◇ Amy Hendrickson
57 Longwood Avenue
Brookline, MA 02446
USA
amyh (at) texnology dot com
http://www.texnology.com