The Wonders of \csname...\endcsname

Amy Hendrickson
57 Longwood Avenue
Brookline, MA 02446
USA
amyh@texnology.com
http://www.texnology.com

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.

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

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.

When the auxiliary file 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, defining the highest number given for that page. Since the definition is made with \csname...\endcsname we can have the page number contained in the name of the definition. This allows us to call the definition in the running head of the LaTeX document, using the current page number in the \csname...\endcsname. We'll see a practical use for this construct.

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

1 The Basics

The TeX primitive commands \csname,\endcsname allow some useful macro constructs that wouldn't otherwise be possible. Here are some of its useful characteristics.

1. We can use \csname to find out if a command has been defined, since an undefined command is equal to \relax. \csname allows us to test to see if this is the case, and make choices based on the result: `\expandafter\ifx \csname anycommand \endcsname\relax <do this>\else<do that>\fi`.

2. Unlike commands made with \def, commands may be defined with \csname...\endcsname that include non-letters in the name of the definition (with the exception of %).

   For example, this is a valid definition that uses symbols and a number in its name: `\expandafter\def\csname $#2\endcsname{Hi!}`. It may be called using \csname,\endcsname, `\csname $#2\endcsname` to produce: Hi!.

3. We can drop macro arguments in the form of a parameter number into \csname...\endcsname. In this example from LaTeX code, \setcounter tests the first argument to see if it is a counter that has been defined with \newcounter; if false, give error message; if true, globally set the counter to be equal to the second argument.

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

   This has the effect of hiding some complexity from the user, who only sees the name of the

counter and the number to which the counter is set, i.e., `\setcounter{page}{21}`.

4. 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 a set of possibilities, we can include a counter in the name of a new definition: `\expandafter\def \csname apple\the\applenum\endcsname{}`

In this article we'll see a number of ways we can use `\csname...\endcsname` with counters.

1.1 Dynamic Macro Building

We can use a counter within `\csname...\endcsname` to make a series of macros, a new one every time the counter is advanced. We do this by including a definition, made with `\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 macro every time it is called.

Using our previous example: `\expandafter \def\csname apple\the\applenum\endcsname` we can make a command that will make more commands in this way:

```
\newcount\applenum
\def\applename#1{\global\advance\applenum by1
\expandafter\def\csname apple\the\applenum
\endcsname{#1}}
```

Now we can access the newly made inner csname macro by using a loop, which advances a counter in each iteration, and calls the csname macro using a counter in the body of its name.

Here we call the newly made csname macros with a loop that advances a counter. This example tests to see if the command is defined, and if true, gives it a number with `\the\loopnum` and calls the command; else, ends the loop.

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

Used:

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

Results:

1. Macintosh
2. Gala

This tool has surprisingly many uses. For our first real world example: making endnotes.

2 Endnotes Example

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 macro every time it is used.

We start by making 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 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 to12pt{\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.

```
\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.
}
```

3 Example: On-line Report Generation

A somewhat similar construction may be used to make hyperlinked tabs for on-line report generation.

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 Tikz 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 the report includes this 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 `\codename` counter in its name, 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 Tikz, here is the 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}}}
```

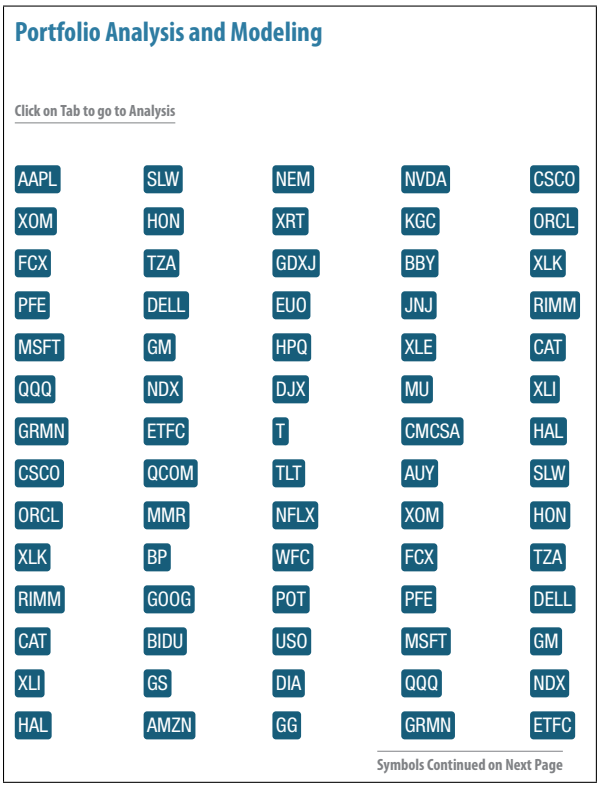

Figure 2: One form of automated online report generation, this is a draft version of customized financial report. Each symbol automatically generated and is hyperlinked to the appropriate page of the report. The company analyzed depends on input from the client; the symbols and their linking is done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

4 Example: Redefinition of macro within the auxiliary file

Our final example shows a technique that might be used to solve various problems. In this case, we are using `\csname` to determine the highest level of security classification on a particular page, so that we can print the highest level on the top and bottom of the page (Unclassified, Classified, Secret and Top Secret). Every paragraph on the page will be marked with one of these classification levels, in any order. Figures and tables and their captions will be marked as well.

This problem is difficult because we don't know initially the page number where each paragraph, table or figure, will appear. In addition we don't have a way of determining which is the highest level for any particular page.

There are many more complications to this general 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, let's consider only the general mechanism here.

The solution

The solution comes from our convenient tool, `\csname`, and its ability to use a counter in its name to make a series of unique definitions.

To this we add the innovation of making further definitions in the auxiliary file using a conditional to determine whether the current classification number is the highest for the current page number. Only if it is the highest number will the pagenumber and the classification level be defined so that the information can later be accessed to be used in the running head and foot for that particular page.

4.1 Setting up

We use a `\write` for every instance where a classifiction 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`

However, we won't open the new auxiliary file in the body of the .cls file, since we need to input the current version of `\jobname.lev` and access the commands found in it before opening a new file.

To do this we can redefine `\document` so that when `\begin{document}` is used it will input the current `\jobname.lev` and only then open the new `\jobname.lev` to be populated with `\csname` commands produced in the next LaTeX run.

We can do this by saving the current version of \document, and then redefine it to use its original definition, plus inputting our auxiliary file, and opening the new file:

```
\let\savedocument\document

\def\document{\savedocument
%% We temporarily open \jobname.lev:
  \openin\@inputcheck\jobname.lev %
%% if end of file=\@inputcheck, no file exists
  \ifeof\@inputcheck
%% if no file let users know they must
%% run LaTeX again:
  \typeout{^^J^^J
  !! Please run LaTeX again to get
  correct classification levels^^J^^J}
\else
%% Since we know now that there Is a file
%% called \jobname.lev we input it and
we get the information from the last
%% LaTeX run:
\input \jobname.lev
\fi
%% And Now we can open the file where info
%% from the current LaTeX run can be saved:
\immediate\openout\collect=\jobname.lev
...
```

4.2 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 case where the beginning of the document is in roman, and the body of the document is in arabic, we 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.

We can use \shipout to advance this counter. (\shipout is the TeX primitive that is called every time a page is completed.) This gives us a number that is continuous through the document.

\shipout will be used to print the classification term on the top and bottom of the page. We use \shipout to generalize this solution, so that this system will work independently of any page style, and its headers and footers.

Now we can add this redefinition of shipout to our redefinition of \document. If the hyperref.sty package has been used, we need to access \shipout through the \AtBeginShipout command, so we test to see if that command is available. If it is not defined, we redefine \shipout directly. However, if \AtBeginShipout is defined, we know that hyperref.sty has been used, and we use \AtBeginShipout to advance the counter and call the classification commands to print the classification level at the top and bottom of the page.

```
\newwrite\collect
\newcount\superpage

\let\savedocument\document

\def\document{\savedocument
  \openin\@inputcheck\jobname.lev %
  \ifeof\@inputcheck
\typeout{^^J^^J
!! Please run LaTeX again to get
correct classification levels^^J^^J}
\else
\input \jobname.lev
\fi
\immediate\openout\collect=\jobname.lev
%%
%% Now test to see if \usepackage{hyperref}
%% has been used:
  \expandafter\ifx
\csname AtBeginShipout\endcsname\relax
%%
%% If hyperref has not been used do this:
%%
\let\saveshipout\shipout
\long\def\shipout\vbox##1{\saveshipout
\vbox{\global\advance\superpage by 1
\vbox to0pt{\vss\topofpage\vskip36pt}
##1\vbox to0pt{\vskip6pt\bottomofpage\vss}
 }}
           \else
%% hyperref is used, so we use \AtBeginShipout
%%
\AtBeginShipout{
\setbox\AtBeginShipoutBox=
\vbox{%% Counter advanced
\global\advance\superpage by 1
%% At top of page:
\vbox to0pt{\vss
\centerline{\classfont\makeclassification}
\vskip12pt}
%% Contents of typeset page:
\box\AtBeginShipoutBox
%% At bottom of page:
\vbox to0pt{%
\centerline{\classfont\makeclassification}
\vss}}}
%% Hyperref error prevention, avoids
%% confusion when secmark is used in a section
%% head and will be be found in a bookmark:
\pdfstringdefDisableCommands{%
\let\uppercase\relax
\let\secmark\secmarkletter}
   \fi} %% end test to see if hyperref is used.
```

4.3 Doing the Writes, the Neat Part!

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 .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 is determined.

The \write sends information to the auxiliary file, \jobname.lev, including several conditional tests. The command looks messy and verbose, when the write is made, since we have to stop the expansion of many commands by preceding each one with \string, except for those 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
%%
%% First test to see if this definition has
%% already been made; if not, do a gdef (global
%% def is necessary for a definition in an
%% auxiliary file that will be input to another
%% file) using csname and the superpage number
%% to define the highest level on that page;
%% If it has been defined, test to see if
%% previous number is lower than previous
%% definition; if so, redefine.
%%
\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 makes more sense visually 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.

4.4 Using the Level Information

As we saw earlier, the \jobname.lev file will be input with \begin{document}. This will bring in a series of definitions, one for each page, that associates a page number with a classification level. We can use this information with every shipout, when the macro \makeclassification will be 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}
```

5 Putting These Techniques to Use

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. To summarize:

We can use `\csname<counter>\endcsname` to generate a new and unique command every time an outer command is used and the counter advanced. For example, an `\endnote` command may be defined that generates a new definition every time it is used.

A `\csname...\endcsname` definition with a counter in its name can be used to generate a series of hypertext targets automatically.

In both of these cases, and in general, we can use a loop, with an internal counter advanced each time it is used, to access the new definitions.

We can stop the loop by testing to see if the most recent `\csname<counter>\endcsname` combination has been defined. Using this method to stop the looping has the advantage that we don't need to know in advance how many definitions were made; we will cycle through all available definitions before ending the loop.

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, based 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.

These techniques add to our understanding of the exceptional flexibility of Knuth's TeX language.

* * *

May your explorations in this territory prove enjoyable and fruitful!