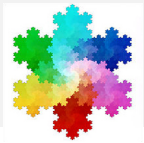
The Joys of Csname

Amy Hendrickson

TeXnology Inc

TeX Users Group



\csname...\endcsname are TEX primitive commands that allow us to define and call macros.

Comparing a definition made with and without csname

1. **Making a definition with** \csname...\endcsname:
   \expandafter\def\csname doggie\endcsname{Toy Poodle}
   and calling the new macro by writing: \csname doggie\endcsname

2. **Making a definition with** \def:
   \def\doggie{Toy Poodle}
   and calling the new macro by writing: \doggie

In either case we will find the results to be 'Toy Poodle'

So, why bother with \csname...\endcsname?

Test to see if command is defined

1. We can use `\csname...\endcsname` to see if a command has been defined. Any term that has not been defined will be equal to `\relax`.

Example

We can make a conditional that tests to see if a term is defined. If defined, do one option, or if false, do the other:

```
\expandafter\ifx\csname FooBar\endcsname\relax
(do this)\else (do that)\fi
```

Use material other than letters in definition name

2. The commands `\csname...\endcsname` allow 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.

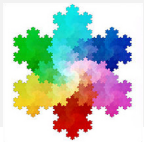
Example

Symbols and numbers in the name of the command

This is a valid definition:
`\expandafter\def\csname $#&\endcsname{Hello!}`
Used: `\csname $#&\endcsname` will produce 'Hello!'

Example

This LaTeX definition uses csname for making macros that might have characters other than letters in their name:
`\def\@namedef#1{\expandafter\def\csname #1\endcsname}`
It is used widely in LaTeX code. For one example, it is used for making labels for cross-referencing, and this is why you can make a label like this: `\label{fig1}`.



Macro argument used within Csname

We can pass a macro argument into \csname...\endcsname

From latex.ltx, the basic LaTeX macro set:

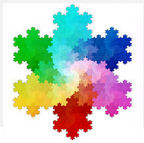
```
\def\setcounter#1#2{\@ifundefined{c@#1}%
  {\@nocounterr{#1}}%
  {\global\csname c@#1\endcsname#2\relax}}
```

Example

Used: \setcounter{page}{201}
The macro checks to see if there is a counter called \c@#1, which in this case, when expanded, is \c@page. 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 to \setcounter{}{}, in this case '201'.

Generic Macro

The example above shows \csname...\endcsname used to make a generic macro call, that will work for any previously defined LaTeX counter.



Commands within csname...endcsname

Here's where things get interesting.

Expand commands within csname

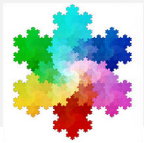
We can expand commands within a definition name made with csname...endcsname. This opens up many complex possibilities.

Example

For one set of possibilities, we can include a counter in the name of a new definition.

```
\expandafter\def\csname apple\the\applenum\endcsname{}
```

In the next set of slides we'll see a number of ways we can use csname...endcsname with counters.



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.

Generating new macros

The counter is advanced in an outer definition, making a new inner definition using the current state of the counter. This has the effect of producing a new and unique macro every time the outer macro is called.

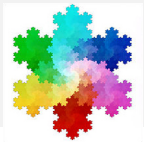
Example

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}}
```



Using Loop to access newly made macros

Everytime we use the \applename macro, we are defining a new macro, named \apple1, \apple2 and so on.

Using Loop to Call Macros

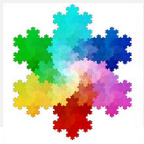
We can access the newly made inner csname macro by using a loop, which advances a counter in each iteration, and calls the csname macro using a counter in the body of its name.

Example

Here we call the newly made csname macros with a loop that advances a counter.

This example tests to see if the command using the current state of the loopnum counter within csname...endcsname has been defined, and if it is, calls the command; else, ends 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
```



Real World Use: Making 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 macro every time it is used.

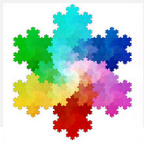
Making endnote definition

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.

Example

```
\newcount\endnum
\def\endnote#1{\global\advance\endnum by 1 $^{\the\endnum}$%
\long\expandafter\def\csname endnote\the\endnum\endcsname{%
\small\leftskip=12pt\relax\parindent=-12pt
\indent\hbox to12pt{\the\loopnum.\hfill}
#1\strut\vskip2pt}}

\let\footnote\endnote % <-- when \footnote{} is written, \endnote{} is used
```



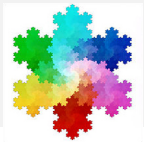
Calling the endnotes

Endnotes

Everytime \footnote{} is used, the argument is saved as the definition of \endnote1, \endnote2, and so on. We access these macros by using a loop that advances a counter each time it is used. The loop will end when it finds an endnote that hasn't been defined.

Example

```
\newcount\loopnum
\def\printendnotes{\global\loopnum=1
\expandafter\ifx\csname endnote\the\loopnum\endcsname \relax
\else\subsection*{Endnotes}\everypar{}
\loop
\expandafter\ifx\csname endnote\the\loopnum\endcsname \relax
\else
\csname endnote\the\loopnum\endcsname % calls the endnote
\vskip2pt
%% Redefine current endnote, set to \relax for next chapter
\global\expandafter\let\csname endnote\the\loopnum\endcsname\relax
\global\advance\loopnum by 1
\repeat
\fi
}
```



Example: On-line Report Generation

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

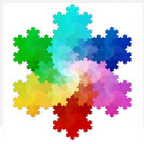
Automating hypertargets

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. The risk analysis for each company will start on a titled new page. Built into the definition for the title of that page is 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, with the stock symbol as its definition, and sends it to the .aux file.



Looping to call csname macros

Example

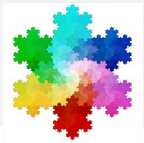
```
\def\maketab#1{\global\advance\codenum by 1
\hypertarget{link\the\codenum}{}
\immediate\write\@auxout{\string\expandafter%
\string\gdef\string\csname\space
tab\the\codenum\string\endcsname{#1}}}
```

Once we have this in place we can use our loop construction for the first, and possibly continuing, pages to build the hyperlinked tabs.

Example

\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}
```



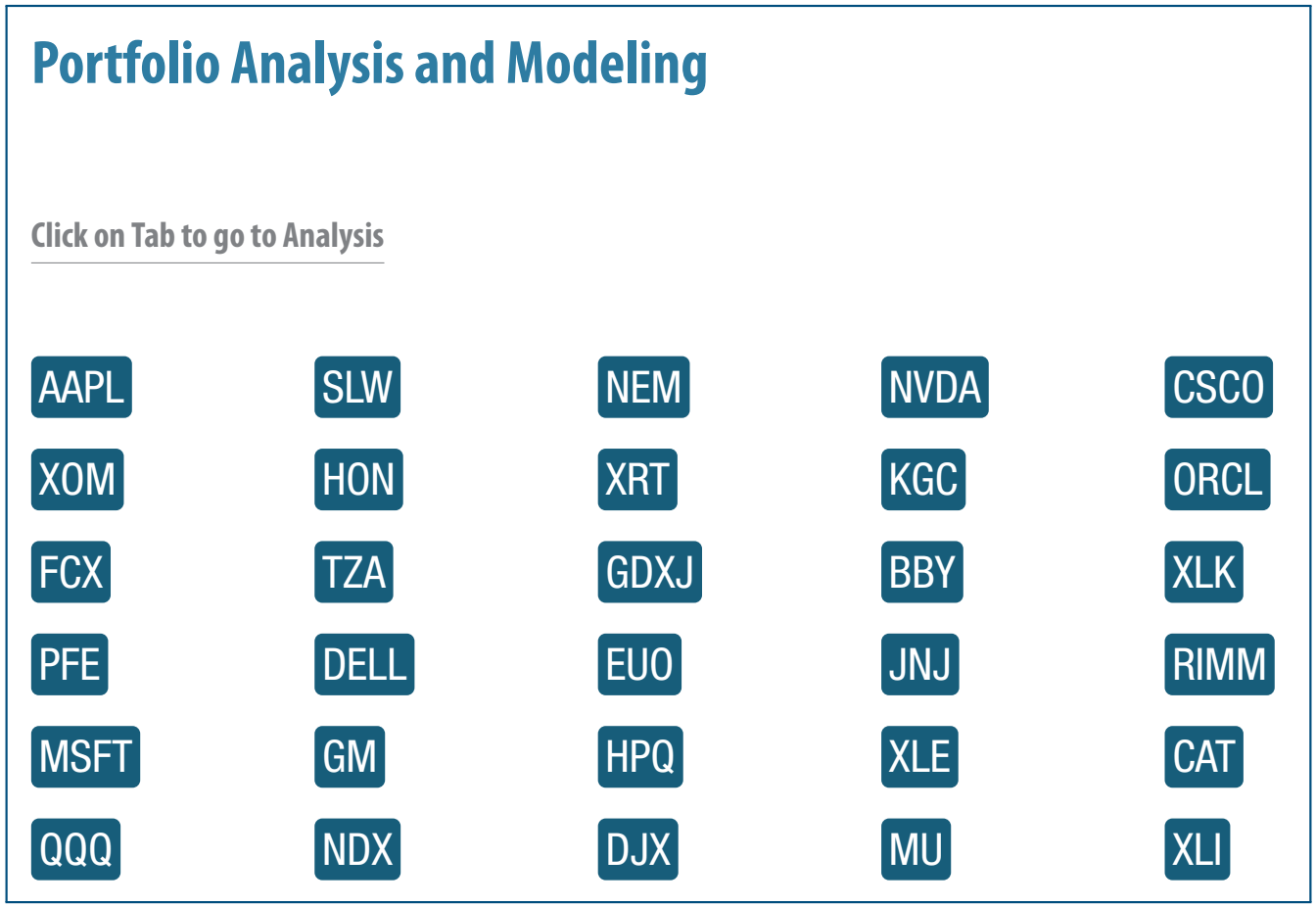
Looping to get the hyperlinks

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

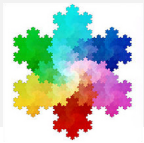
Definition of gettabs

```
\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, below, is made using Tikz
{\plaintab{\csname tab\the\loopnum\endcsname}
\hskip12pt}
\hfill}%% <== end \hbox started above
\global\advance\loopnum by 1
            \repeat}
```

The draft version of the resulting report looks like this:



with each symbol linked to the page containing the risk analysis report on that particular stock.



Csname used for Tabbed Documentation

A similar technique is used for making hyperlinked tabbed documentation:

WILEY-INTERSCIENCE

Using LaTeX for Typesetting Wiley Books

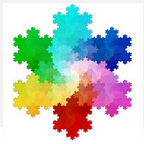
Welcome | Getting Started | Edited Book | Front | Chapters | Graphics | **Figs/Tables** | Example/More | Probs/Exers | Solutions | End | Sci WP

Figures | Tables | Table Notes | Special Captions | **Figure and Table Tips**

To Rotate figure or table

You need `\usepackage{graphicx}`, and then you can use `\rotatebox{(angle)}{\vbox{ table or figure }}`, ie.,

```
\begin{table}[p]
\rotatebox{90}{\vbox{
\caption{This is the table caption.}
\begin{tabular}{crccc}
\multicolumn{3}{l}{\bf Parameters}&\\
\end{tabular}}}
\end{table}
```



Another Csname technique, used for Classification Levels

The Problem

The highest level of classification on any particular page must appear at the top of a classified document.

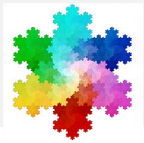
When the classification level is given, the user doesn't know what page it will appear on, and doesn't know in advance whether this particular marking is the highest on the page.

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 on the page**. When the auxiliary file is input, the definition of the highest level on the page is defined.



Code in auxiliary file

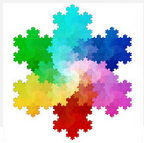
(For the complete code, please see my paper in the TUG conference proceedings.)

Example

Here is what the code looks like when it appears in the auxiliary file:

```
\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`.



Definitions in .aux file input

The auxiliary file will be automatically input into the root .tex file using a redefinition of `\begin{document}`. This will bring in a series of definitions, one for each page, that associates a page number with a number for the highest classification level on that page.

Example

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 where the newly defined LevelOnSuperPage(number) is used:

```
\def\makeclassification{%
...
\centerline{%
\ChangeNumIntoClassification{\expandafter\csname LevelOnSuperPage\the\superpage
\endcsname}}...
```

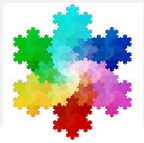
Change Number into Classification Term

The last part needed is to change the number (from 1 to 4) produced when `\expandafter\csname LevelOnSuperPage\the\superpage` is called.

We can do this with the macro `\ChangeNumIntoClassification`. It 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}
```

Now the highest level of classification will print at the top of the 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, ie, cross referencing label.
3. Making a generic macro, that can be modified with with the argument of another macro.
4. Generating new macros by using a counter within csname...endcsname;
5. Calling macros made with csname in the body of a loop. Stopping the loop by testing to see if 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.
6. A csname...endcsname definition with 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, 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.

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!

– Amy

Amy Hendrickson
http://www.texnology.com
amyh@texnology.com

(Slides will be available at
http://www.texnology.com/talk.pdf)