LaTeX, A Short Course
Custom Commands (Macros)

Mark G. Eramian

The University of Saskatchewan

August 29, 2008
Creating your own commands: Macros

One of the more powerful features of \LaTeX{} is that users can define their own commands, much like one defines one’s own procedures in a programming language.

This can be extremely useful for

- Shortening oft-used built-in commands to save typing.
- Giving a special name to symbols or strings for easy global replacement.
- Automating typesetting of similar, but slightly different portions of your document.
Custom Commands

- Custom commands are created with the `\newcommand` command.
- This command has the following syntax:

  \[ \texttt{newcommand}\{\texttt{command\_name}\}[narg][opt]\{def\} \]

  - \textit{narg} Number of arguments to the command (optional)
  - \textit{opt} Specify optional arguments (optional)
  - \textit{def} The definition of the new command.

- When the command is invoked, the command “evaluates” to the source that is built by \textit{def}.
Commands without arguments are useful for defining a “shorthand” for a frequently used sequence \LaTeX commands, text, and symbols.

Example: typesetting a backslash is rather cumbersome:
\$\backslash\$.

In preparing these slides, I defined my own command to simplify this:
\texttt{\newcommand{\bsl}{$\backslash$}}

Now I can typeset a backslash using just \bsl.
Example: Commands without arguments

- In this example we simplify the definition of an indexed sequence:
  \texttt{\newcommand{\xvec}{$x_1, \ldots, x_n$}}
- Now \texttt{xvec} typesets as \textit{$x_1, \ldots, x_n$}.
A Problem

Recall our last definition:
\newcommand{\xvec}{$x_1, \ldots, x_n$}

This definition suffers from the problem that it cannot be used in math mode!

Reason:
\$\xvec\$ expands to \$x_1, \ldots, x_n\$

Now we have two empty math environments with our formula in between.

This will cause \LaTeX{} to issue an error because we have math commands outside of a math environment.
Problem is solved by using the \ensuremath{} command.

If \ensuremath appears in text mode, it automatically switches to math mode before commands in its argument are typeset.

Revising our definition:
\newcommand{\xvec}{\ensuremath{x_1, \ldots, x_n}}

Now both $\xvec$ and \xvec are allowed and produce the same result.
Suppose we want to extend our definition of \texttt{xvec} to allow the indices of the $x$'s to be specified.

We need to allow two arguments to be given to \texttt{xvec}. This is done by using the optional \texttt{nargs} argument to \texttt{newcommand}.

We once again revise our definition of \texttt{xvec}:

\begin{verbatim}
\newcommand{\xvec}[2]
  \ensuremath{x_{#1},\ldots,x_{#2}}
\end{verbatim}

The \#1 and \#2 in the definition get replaced by arguments 1 and 2 to \texttt{xvec} respectively.
Example

\[ \text{\texttt{xvec}\{2\}\{6\}} \] typesets as \( x_2, \ldots, x_6 \)
\[ \text{\texttt{xvec}\{(a,b)\}\{(c,d)\}} \] typesets as \( x(a,b), \ldots, x(c,d) \)
\[ \text{\texttt{xvec}\{y_1\}\{y_2\}} \] typesets as \( y_1, \ldots, y_2 \)
1. Define a command that typesets its argument as both bold and italic.

2. Define a command that takes two arguments:
   a) a font sizing command (e.g. \texttt{\small})
   b) a line of text

and define the command so that the text in the second argument is centered and is sized according to the first argument.
A common pitfall

A common problem that occurs with user-defined macros with no arguments is illustrated as follows:

- Consider the definition
  \newcommand{\ill}{I like \LaTeX}

- If we try to typeset
  \ill because it looks pretty.

  then we will get:

  I like \LaTeX because it looks pretty.

- \LaTeX assumes that the space between \ill and because is only to denote the end of the command name and is not a real space.
A common pitfall

To solve the problem we force a space as follows:
\ill\ because it looks pretty.
which typesets as
I like $\LaTeX$ because it looks pretty.
Additional Customization

- It is possible to:
  - create custom commands with optional arguments
  - create custom text environments
  - create custom theorem environments
  - create your own packages
  - create your own document class files

- Unfortunately all of these things are well beyond the scope of this workshop.