Commands and Figures

William Beason and Evan Ott

March 5, 2014

Table of Contents

Table of Contents

Requirements for Commands

Commands need

Requirements for Commands

Commands need

- a name so they can be called, and

Requirements for Commands

Commands need

- a name so they can be called, and
- something to be replaced with.

Requirements for Commands

Commands need

- a name so they can be called, and
- something to be replaced with.

To declare a new command, use the `\newcommand` command. It has two required arguments.

Requirements for Commands

Commands need

- a name so they can be called, and
- something to be replaced with.

To declare a new command, use the `\newcommand` command. It has two required arguments.

```
\newcommand{\name}{replacement}
```

Requirements for Commands

Commands need

- a name so they can be called, and
- something to be replaced with.

To declare a new command, use the `\newcommand` command. It has two required arguments.

```
\newcommand{\name}{replacement}
```

The above command makes it so whenever LaTeX comes across `\name`, it replaces it with `replacement`.

Example Command

Say you need to display backslashes often in a $\LaTeX$ document or Beamer presentation (like this one). Typing `\textbackslash` quickly becomes tiresome.

Example Command

Say you need to display backslashes often in a LaTeX document or Beamer presentation (like this one). Typing `\textbackslash` quickly becomes tiresome.

Intsead, try:

```
\newcommand{\tb}{\textbackslash}
```

Now whenever you need to reference the backslash, just type `\tb`. LaTeX will now replace every instance of `\tb` with `\textbackslash`.

Arguments

Arguments add more functionality to commands by allowing variable replacement text.
Remember to

Arguments

Arguments add more functionality to commands by allowing variable replacement text.
Remember to

- specify number of arguments as the optional argument, and

Arguments

Arguments add more functionality to commands by allowing variable replacement text.
Remember to

- specify number of arguments as the optional argument, and
- reference arguments within the replacement with poundsign-number (e.g. #2).

Normally, commands can take a maximum of nine arguments.

Argument Example

For example in,

```
\newcommand{\pfrac}[2]{\left(\frac{#1}{#2}\right)}
```

⋮

```
\pfrac{1}{2}
```

the second line is replaced with

```
\left(\frac{1}{2}\right)
```

and produces

$$\left(\frac{1}{2}\right)$$

Environment Commands

Sometimes entering and exiting environments can be tiresome, especially if they are used in mostly the same way every time. For example, `pmatrix`. Commands can specify entering and exiting the environment, and the required arguments can be what goes inside.

Environment Commands

Sometimes entering and exiting environments can be tiresome, especially if they are used in mostly the same way every time. For example, `pmatrix`. Commands can specify entering and exiting the environment, and the required arguments can be what goes inside.

```
\newcommand{\epmatrix}[1]{\begin{pmatrix}#1\end{pmatrix}}
```

⋮

```
\epmatrix{1&0\\0&1}
```

Produces

$$\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$$

Optional Arguments

Optional arguments are present in many LaTeX commands. To make the square root sign into a cube root sign, add [3] before the required argument.

Optional Arguments

Optional arguments are present in many LaTeX commands. To make the square root sign into a cube root sign, add [3] before the required argument.

So

```
\sqrt[3]{2}
```

Produces

$$\sqrt[3]{2}$$

Creating Optional Arguments

To define an optional argument, use the *second* optional argument for `\newcommand`. The optional argument is *always* the first argument.
Say we want the integral to automatically designate the variable of integration. We could make it by default choose x, and allow it to bet set to anything else.

Creating Optional Arguments

To define an optional argument, use the *second* optional argument for `\newcommand`. The optional argument is *always* the first argument.

Say we want the integral to automatically designate the variable of integration. We could make it by default choose x, and allow it to bet set to anything else.

```
\newcommand{\intd}[1][x]{\int \text{d}#1\,}
```

```
\intd x^2
\intd[y] y^2
\intd[] a
```

```
\intd x^2
\intd[y] y^2
\intd[] a
```

Produces

```
\intd x^2
\intd[y] y^2
\intd[] a
```

Produces

$$\int \mathrm{d}x\, x^2$$

$$\int \mathrm{d}y\, y^2$$

```
\intd x^2
\intd[y] y^2
\intd[] a
```

Produces

$$\int \mathrm{d}x\, x^2$$
$$\int \mathrm{d}y\, y^2$$

If we wanted the optional argument to default to nothing, we need to declare the optional argument with $[\{\}]$.

```
\newcommand{\intd}[1][{}]{\int \text{d}#1\,}
```

Renewing Commands

Sometimes you want to redefine base commands. To do this, use `\renewcommand`.

Say we want to format sections so they use roman numberals instead of arabic.

```
\renewcommand{\thesection}{\Roman{section}}
```

Errors

Never

Errors

Never

- use anything but alphabetic characters in command names,

Errors

Never

- use anything but alphabetic characters in command names,
- define a command which has already been defined,

Errors

Never

- use anything but alphabetic characters in command names,
- define a command which has already been defined,
- use a command within its own definition (or redefinition), or

Errors

Never

- use anything but alphabetic characters in command names,
- define a command which has already been defined,
- use a command within its own definition (or redefinition), or
- create a closed loop.

Table of Contents

Packages

- `graphicx` - required, allows images to be loaded
- `caption` - options to customize captions (but not required to have captions)
- `subcaption` - captions on subfigures
- `placeins` - defines `\FloatBarrier`
- `float` - not required for floats, but required for precise positioning

Commands and Environments

Commands

- `\includegraphics` - allows the insertion of graphics
- `\caption` - text at the bottom of a figure
- `\label` - label to reference the figure by
- `\FloatBarrier` - a barrier floats shall not pass

Environments

- `figure`
- `wrapfigure`
- `subfigure`
- `tabular`
- `table`

The \includegraphics Command

To include a graphic, use `\includegraphics{image.jpg}`

The \includegraphics Command

To include a graphic, use `\includegraphics{image.jpg}`

With `graphicx` when compiling with pdflatex (you should be using this already), you can import `.jpg`, `.png`, and `.pdf`, and sometimes `.eps`. Some installations do not natively support `.eps`, so if want to import them, load the `epstopdf` package *after* `graphicx`.

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

- `[width=xx]` and `[height=xx]` specify the preferred width and height of the image. If only one is specified, the aspect ratio is maintained

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

- `[width=xx]` and `[height=xx]` specify the preferred width and height of the image. If only one is specified, the aspect ratio is maintained
- `[keepspectratio=xx]` forces the image to maintain the aspect ratio even if both `width` and `height` are defined - makes sure neither is exceeded

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

- `[width=xx]` and `[height=xx]` specify the preferred width and height of the image. If only one is specified, the aspect ratio is maintained
- `[keepspectratio=xx]` forces the image to maintain the aspect ratio even if both `width` and `height` are defined - makes sure neither is exceeded
- `[angle=xx]` sets the angle in degrees the image is rotated counter-clockwise

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

- `[width=xx]` and `[height=xx]` specify the preferred width and height of the image. If only one is specified, the aspect ratio is maintained
- `[keepspectratio=xx]` forces the image to maintain the aspect ratio even if both `width` and `height` are defined - makes sure neither is exceeded
- `[angle=xx]` sets the angle in degrees the image is rotated counter-clockwise
- `[trim=l b r t]` crops the image by specified measurements from the left, bottom, right, and top

Manipulating Graphics

There are several options for manipulating a graphic when it is imported.

- `[width=xx]` and `[height=xx]` specify the preferred width and height of the image. If only one is specified, the aspect ratio is maintained
- `[keepspectratio=xx]` forces the image to maintain the aspect ratio even if both `width` and `height` are defined - makes sure neither is exceeded
- `[angle=xx]` sets the angle in degrees the image is rotated counter-clockwise
- `[trim=l b r t]` crops the image by specified measurements from the left, bottom, right, and top
- `[clip=true]` allows the image to be trimmed - without it, `trim` does nothing

Resizing Example

Note that LaTeXapplies width and height *before* rotation, so `width=1in, height=2in, angle=90` will produce an image which actually has width 2 and height 1.

See example.

Floats

Float environments are containers for things which should not be broken across pages. The predefined ones are `table` and `figure`.

`table` is mainly used as a wrapper for `tabular`, allows for tables to be created.

`figure` can contain almost anything - useful if you want to add a caption to equations

Creating a Float

```
\begin{figure}
\begin{center}
\includegraphics[scale=0.9]{waterfall.jpg}
\end{center}
\caption{A waterfall at Yosemite.}
\end{figure}
```

See float example.

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

- [h] suggests to put the figure approximately where the figure appears in the code

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

- [h] suggests to put the figure approximately where the figure appears in the code
- [t] and [b] put the figure at the top or bottom of whichever page LaTeXdecides to put the figure in

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

- [h] suggests to put the figure approximately where the figure appears in the code
- [t] and [b] put the figure at the top or bottom of whichever page LaTeXdecides to put the figure in
- [p] puts the figure in a special page with just figures

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

- [h] suggests to put the figure approximately where the figure appears in the code
- [t] and [b] put the figure at the top or bottom of whichever page LaTeXdecides to put the figure in
- [p] puts the figure in a special page with just figures
- [!] tells LaTeXto not care as much about calculating

Float Positioning

While images not in the `figure` environment stay in the same place, LaTeX calculates optimal places to position floats. There are several ways to modify the default behavior with the optional argument to the figure environment.

- [h] suggests to put the figure approximately where the figure appears in the code
- [t] and [b] put the figure at the top or bottom of whichever page LaTeXdecides to put the figure in
- [p] puts the figure in a special page with just figures
- [!] tells LaTeXto not care as much about calculating
- [H] puts the float at the exact position in the code (not a suggestion like [h!]) - requires the `float` package

\FloatBarrier

LaTeXtakes suggestions rather lightly and has strong opinions about where figures should be placed, so sometimes *it just won't listen*. To counter this, load the `placeins` package and use the `\FloatBarrier` command.

`\FloatBarrier` creates a barrier which floats cannot cross. If you place one before and after a float in the code, LaTeXis forced to typeset the float in that precise position.