Lecture 7 - Additional Packages Info and User-Definables

LaTeX for Math and Science
Christian Blanco and Brandon Eltiste

Spring 2011 Lecture 7

Outline

Adding local tex folder

TeXLive and MikTeX have the capability of letting users place packages outside of the installation paths. This is convenient when neither programs install the packages automatically. Also, in Windows you do not need to run a program as administrator, which is the most compelling reason to do this. There is also less digging for the actual folder or file.

- TeXLive Instructions
- MikTeX Instructions

Note that you only need to run texhash after the first time you add a package, and not each time you edit the package. Also, the folder you point as a root must contain `tex\latex`, which is where you place packages, you can place them in their own folders or just the `latex` folder.

Who cares?

- What if a math command like sin, `\sin` is not defined for your operator, e.g. `\dom`, `\ran`, or `\Res`?
- What if I am too lazy to write out the whole command?
- What if I don’t like the look of the existing command?

Who cares?

- What if a math command like sin, `\sin` is not defined for your operator, e.g. `\dom`, `\ran`, or `\Res`?
- What if I am too lazy to write out the whole command?
- What if I don't like the look of the existing command?

Who cares?

- What if a math command like sin, `\sin` is not defined for your operator, e.g. `\dom`, `\ran`, or `\Res`?
- What if I am too lazy to write out the whole command?
- What if I don't like the look of the existing command?

Who cares?

- What if a math command like sin, `\sin` is not defined for your operator, e.g. `\dom`, `\ran`, or `\Res`?
- What if I am too lazy to write out the whole command?
- What if I don't like the look of the existing command?

DeclareMathOperator

So LaTeX doesn't have `\dom` or `\ran`, built-in, so that is what `\DeclarMathOperator` is for.
Simply place `\DeclareMathOperator{\dom}{dom}` and `\DeclareMathOperator{\ran}{ran}` in the preamble on their own lines.
Now you can use `\dom x` and `\ran x`, which look like $\operatorname{dom} x$ and $\operatorname{ran} x$ respectively.
The importance is that `\DeclareMathOperator` makes the text upright and adds a small space after the command.

DeclareMathOperator$*$

So the default behavior for subscripts and superscripts for `\DeclareMathOperator` is ran_x and dom^x.
What if I want it to be like `\sum`, i.e.

$$\sum_n^\infty \quad ?$$

Use `\DeclareMathOperator*`. For example, `\DeclareMathOperator*{\Res}{Res}`. So it will appear as

$$\operatorname*{Res}_{z=0} \frac{1}{z} \quad ?$$

The similarities between newcommand, renewcommand, and providecommand

These commands each have the same declarations:

-
  ```
  \*command{\Name}[NumberOfArguments]{The command}
  ```

A couple examples

- `\newcommand{\bbR}{\ensuremath{\mathbb{R}}}` results in $\mathbb{R}$ when you type `$\bbR$`.

-
  ```
  \newcommand{\dotssub}[3]{\ensuremath{{#1}_{#2}
  ```
  results in $a_1 a_n$ when you type `\dotssub{a}{1}{n}`.

-
  ```
  \newcommand{\Iff}{if\textcompwordmark f\xspace}
  ```
  results in iff when you type `\Iff`.

The similarities between newcommand, renewcommand, and providecommand

These commands each have the same declarations:

- `\*command{\Name}[NumberOfArguments]{The command}`

A couple examples

- `\newcommand{\bbR}{\ensuremath{\mathbb{R}}}` results in $\mathbb{R}$ when you type `$\bbR$`.
- `\newcommand{\dotssub}[3]{\ensuremath{{#1}_{#2}` results in $a_1 a_n$ when you type `\dotssub{a}{1}{n}`.
- `\newcommand{\Iff}{if\textcompwordmark f\xspace}` results in iff when you type `\Iff`.

The similarities between newcommand, renewcommand, and providecommand

These commands each have the same declarations:

- `\*command{\Name}[NumberOfArguments]{The command}`

A couple examples

- `\newcommand{\bbR}{\ensuremath{\mathbb{R}}}` results in $\mathbb{R}$ when you type `$\bbR$`.
- `\newcommand{\dotssub}[3]{\ensuremath{{#1}_{#2}` results in $a_1 a_n$ when you type `\dotssub{a}{1}{n}`.
- `\newcommand{\Iff}{if\textcompwordmark f\xspace}` results in iff when you type `\Iff`.

The similarities between newcommand, renewcommand, and providecommand

These commands each have the same declarations:

- `\*command{\Name}[NumberOfArguments]{The command}`

A couple examples

- `\newcommand{\bbR}{\ensuremath{\mathbb{R}}}` results in $\mathbb{R}$ when you type `$\bbR$`.
- `\newcommand{\dotssub}[3]{\ensuremath{{#1}_{#2}` results in $a_1 a_n$ when you type `\dotssub{a}{1}{n}`.
- `\newcommand{\Iff}{if\textcompwordmark f\xspace}` results in iff when you type `\Iff`.

The similarities between newcommand, renewcommand, and providecommand

These commands each have the same declarations:

- `\*command{\Name}[NumberOfArguments]{The command}`

A couple examples

- `\newcommand{\bbR}{\ensuremath{\mathbb{R}}}` results in $\mathbb{R}$ when you type `$\bbR$`.
- `\newcommand{\dotssub}[3]{\ensuremath{{#1}_{#2}` results in $a_1 a_n$ when you type `\dotssub{a}{1}{n}`.
- `\newcommand{\Iff}{if\textcompwordmark f\xspace}` results in iff when you type `\Iff`.

The differences between newcommand, renewcommand, and providecommand

- `\newcommand` will stop if the command has been declared before by another package. `\newcommand` will return an error if it does find it has already been declared.
- `\renewcommand` will overwrite any command declared before, and not warn you that it overwrote it. Be careful with this command.
- `\providecommand` is like newcommand, but doesn't warn you if it has been declared before. This is the safest to use, and should be used most often.

The differences between newcommand, renewcommand, and providecommand

- `\newcommand` will stop if the command has been declared before by another package. `\newcommand` will return an error if it does find it has already been declared.
- `\renewcommand` will overwrite any command declared before, and not warn you that it overwrote it. Be careful with this command.
- `\providecommand` is like newcommand, but doesn't warn you if it has been declared before. This is the safest to use, and should be used most often.

The differences between newcommand, renewcommand, and providecommand

- `\newcommand` will stop if the command has been declared before by another package. `\newcommand` will return an error if it does find it has already been declared.
- `\renewcommand` will overwrite any command declared before, and not warn you that it overwrote it. Be careful with this command.
- `\providecommand` is like newcommand, but doesn't warn you if it has been declared before. This is the safest to use, and should be used most often.

The differences between newcommand, renewcommand, and providecommand

- `\newcommand` will stop if the command has been declared before by another package. `\newcommand` will return an error if it does find it has already been declared.
- `\renewcommand` will overwrite any command declared before, and not warn you that it overwrote it. Be careful with this command.
- `\providecommand` is like newcommand, but doesn't warn you if it has been declared before. This is the safest to use, and should be used most often.

Keep the following in mind

- When you want to change an existing, do `+, rather than .`
- `These can make your tex+ file easier to read for yourself, but` <u>`not`</u> `for others.`
- `When sharing do one of the following:`
  - `Make sure the commands declared are in the preamble.`
  - `OR do a find and replace for the commands declared.`

Keep the following in mind

- When you want to change an existing, do `+, rather than .`
- `These can make your tex+ file easier to read for yourself, but` <u>`not`</u> `for others.`
- `When sharing do one of the following:`
  - `Make sure the commands declared are in the preamble.`
  - `OR do a find and replace for the commands declared.`

Keep the following in mind

- When you want to change an existing, do +, `rather than .`
- `These can make your tex+ file easier to read for yourself, but` <u>`not`</u> `for others.`
- `When sharing do one of the following:`
  - `Make sure the commands declared are in the preamble.`
  - `OR do a find and replace for the commands declared.`

Keep the following in mind

- When you want to change an existing, do `+, rather than .`
- `These can make your tex+ file easier to read for yourself, but` <u>`not`</u> `for others.`
- `When sharing do one of the following:`
  - `Make sure the commands declared are in the preamble.`
  - `OR do a find and replace for the commands declared.`

Keep the following in mind

- When you want to change an existing, do `+, rather than .`
- `These can make your tex+ file easier to read for yourself, but` <u>`not`</u> `for others.`
- `When sharing do one of the following:`
    - `Make sure the commands declared are in the preamble.`
    - `OR do a find and replace for the commands declared.`