examtex documentation

Dhruva Karkada

Nov 27 2018

Contents

1 Overview

examtex is a tool for building pretty-looking exams from a simple markup file. This is done by converting the markup file into a LaTeX (file extension .tex) document. This requires Python 3. Then, the tex needs to be compiled into a pdf; this requires an installation of TeX. An alternative would be to input the tex file into an online service such as Overleaf, and have them create the pdf. For those using bash, this tool also provides a bash script which is capable of automatically generating both the tex and the pdf (given a working installation of both Python 3 and `latexmk`).

The first step of the compilation process can easily be done from the terminal using

```
python3 examtex.py MyExam.exam
```

where `MyExam.exam` is the markup file you wrote. This command will run the program `examtex.py`, which generates two files: the .tex file for the exam, and the .tex file for the answer key. In the second step of the compilation process, both .tex files can be compiled into pdfs using any TeX distribution (typically using the program `latexmk`). Or, as mentioned before, you could use an online service.

For convenience, I've written the python terminal script `examtex` which can do both compilation steps at once:

```
./examtex [-p][-c] MyExam.exam
```

You will need to have `latexmk` and Perl installed (you should be fine if you've installed any TeX distribution). The script takes optional flags: `-p` will convert the tex to pdf, and `-c` will both convert the tex to pdf and clean up the auxiliary files after the pdfs have been created.

2 Making .exam markup files

The markup file uses a simple hierarchical syntax for defining parts of an exam. At the top level, there are sections. These define different sections of the exam, mainly for semantic purposes. Sections are made up of modules, which functionally define parts of the exam (a module for the section title, a module for a true/false section, a module for instructions, etc.). Finally, within modules we have bangs: short commands which set options or insert a small item which doesn't necessarily require its own module.

Sections and modules are both declared using the same syntax: `[keyword]`, where *keyword* is the type of section or module you want to have. Bangs are preceded by an exclamation mark, and may contain arguments contained in curly braces: `!bang {arg1, arg2}`.

Unlike the three varieties of sections, there are many more types of modules; some of them are special to the Header and Cover section, some can only be used in the Section sections, while others can be used in any type of section. They will be discussed in detail later.

Bangs are preceded by an exclamation mark, and may contain arguments contained in curly braces: `!bang {arg1, arg2}`. Often, arguments are either flags or key-value (KV) pairs.

2.1 Sections

There are three types of sections: Header, Cover, and Section. The Header and Cover are special sections; there can be at most one of each, and they must be at the beginning of the exam. On the other hand, Section is the general-purpose section, and acts as a semantic marker between parts of the exam.

2.1.1 Header

The Header section acts as a sort of metadata section. If you include a header, it must be the first section you define. Here, you can define which LaTeX packages to import using the `!pkg` bang. In addition, you can specify three text items to put in a header above each page (left, center, and right). To leave one of them empty, just leave a double-slash in that line.

```
[Header]
!pkg {physics, hyperref}
//
Exam title
Student ID
```

2.1.2 Cover

This defines the cover page for the exam. If you include a cover page, it must be the first section you define (other than the header, which takes precedence). The only valid modules to use in the cover page are: Title, Subtitle, Author, Info, Text, Latex, and Table. All the bangs are valid to use inside the Cover section (although some of them may do nothing). More details can be found later in this document.

2.1.3 Section

This marks general sections within the exam. It's important to have at least one Section to mark the end of the Header or Cover and the beginning of the exam. New Sections automatically generate a pagebreak. The only valid modules to use are: Title, Text, Latex, Table, Match, TF, MC, and FRQ. All bangs are valid. More details can be found later in this document.

2.2 Modules

There are 11 types of modules. 4 of them (Title, Text, Latex, Table) can be in any section; 3 of them (Subtitle, Author, Info) can be used only in the Cover; and the other 4 (Match, TF, MC, and FRQ) represent question types and can only be used in the actual body of the exam.

Declaring a module uses the same syntax as declaring a section: `[Module]`. After this line, all the content underneath belongs to this module, until the next module or section declaration. Different modules have syntaxes, which are described below.

2.2.1 Title

Takes a single line of text and makes it big. On the Cover, it makes the title of the exam, whereas in a section, it is typically used to label the section.

```
[Title]
Part II: Interpreting Data
```

2.2.2 Text

Make paragraphs of text. Useful for writing instructions on the cover page or for a section. Use `\i{stuff}` and `\b{stuff}` for italics and bold. Simple LaTeX can be interpreted, such as the `noindent` command, as well as math mode.

```
[Text]
Lorem Ipsum.  \b{bold text} and math $y=e^{e^t}$.  First
paragraph unindented as per \LaTeX convention.
Here's a new paragraph, automatically indented.
```

2.2.3 Latex

Directly insert LaTeX into the document.

```
[Latex]
\\[.2 in]
\par
\begin{itemize}
\item item 1
\item item 2
\end{itemize}
```

2.2.4 Table

Simple table formatting. Use tabs to separate items in each row; newline to separate rows. Use a line of just dashes to insert a horizontal line. Use `\i{stuff}` and `\b{stuff}` for italics and bold.

The `!options` bang is required for Table, and it must contain the `pattern` KV pair. The pattern uses the syntax of tabular, the LaTeX package, to specify the number of columns, the positions of column separators, and the text alignment in each column. Refer to the tabular documentation for more details. Other potential options for table formatting are the `boxed` flag (which frames the table), and `linespace` KV pair (which specifies the spacing between rows).

```
[Table]
!options {pattern=c|c, linespace=1.25, boxed}

\b{Age}     \b{Height}
----------------
11     5'2"
13     5'3"
14     5'6"
16     5'8"
18     5'9"
```

2.2.5 Subtitle

A subtitle for the cover page.

```
[Title]
Introduction to Topology
[Subtitle]
Midterm 2
```

2.2.6 Author

Specifies the author(s) of the exam, along with potential contact information, separated by a comma. Cover page only. The name is bolded and the contact information is italicized.

```
[Author]
Dhruva Karkada, dkarkada@gmail.com
Another Author, contact@gmail.com
```

2.2.7 Info

Provides a place on the Cover for the student to write his information. Each line represents a labelled blank line for the student to fill out.

```
[Info]
Student Name
Date
Any other info
```

2.2.8 Match

A module for having a matching (wordbank) section on the exam. The wordbank can have up to 26 items in it. Questions are added using the syntax `Answer::Question`. Answers used more than once will only appear in the wordbank once. To add an answer to the wordbank without a corresponding question (i.e. an extraneous answer choice), use a doubleslash `//` in place of the question.

Answers are sorted alphabetically in the wordbank. Questions are automatically shuffled, unless you include the flag `noshuffle` in the `!options` bang. To print the point value of each question, include the KV pair `qworth` in the options (note that this only supports a single point value for all the questions in this module).

```
[Match]
!options {qworth=2, noshuffle}
Answer 1::  This is question 1
Answer 2::  This is question 2
Extra ans::  //
```

2.2.9 TF

True/False questions. The syntax is exactly the same as for match, except the only allowable answers are T and F. No wordbank is created. The `noshuffle` and `qworth` options remain.

```
[TF]
!options {noshuffle}
F::  Mars is blue
F::  America was founded in 1634
T::  examtex is really cool and useful
```

2.2.10 MC

Multiple choice questions. The first line of a question contains the question itself, which is followed by a list of indented lines containing the answer choices. Indents must be done with a tab, not spaces. Questions are not shuffled. There is support for simple LaTeX, math mode (`$$ math $$`), as well as the italics and bold commands (`\i{}` and `\b{}`).

The correct answer should be the first choice listed; the generated question will have the choices shuffled randomly. If you want to write a question where the answers are not shuffled, mark the correct answer by appending the `{C}` tag. Upon detecting this tag, the parser will know not to shuffle the answer choices.

A helpful option to provide is the `intro-height` KV pair. Guess the height of the pre-MC contents on the page (such as the title and instructions) so the program can account for that when determining spacing. It's better to overestimate this value than to underestimate. Other useful options are the `twocolumn` flag (which makes two columns of MC questions) and the `qworth` KV pair (explained in the Match module documentation).

```
[MC]
!options {intro-height=30, twocolumn}
Bubble sort runs in
    Polynomial time
    Constant time
    Exponential time
    Logarithmic time
Red-black trees were invented in
    1952
    1962
    1972 {C}
    1982
Which is \b{NOT} true about $$ f'(x) $$?
    It represents the concavity of the function
    It is the derivative of the function
    It equals zero at local maxima
    It is a function
```

2.2.11 FRQ

The most versatile question type: free response questions. The FRQ module is very versatile; it can support parts and subparts, question-specific point values, and question-specific heights for the student solution space. Just like other modules, it supports math mode and simple LaTeX commands, as well as the bold and italics aliases mentioned earlier.

Questions are unindented; its parts are indented once, and a part's subparts are indented once more. If you want to have an empty question with parts, put a single asterisk in the line for the empty question. To specify the point value for a (sub)question, put the value in curly braces before the question.

The solution to any question or subquestion is specified directly underneath its (sub)question, indented once (relative to its (sub)question), and preceded by a double slash. Using the `!ans-options` bang, you can specify whether there's an answer sheet for the student to write the answer; by default, there'll be space left underneath every question (a more detailed explanation can be found later in the documentation). The amount of space the student gets can be specified by putting the number of lines the answer should be in curly braces in front of the solution (but after the double slash); if unspecified, the program will guess based on the length of the provided solution.

```
[FRQ]
{10} A question worth 10 points
    {8} Here is a part
        // Here is its solution
    Here is another part
        {1} Its first subpart
            // and respective solution
        {1} Next subpart
            // {10} and 10-line-long solution
*
    This question has no high-level question
        // nice
    It only shows these two question parts
        // ok
```

2.3 Bangs

Bangs are short commands which can take arguments (enclosed in curly braces). Most bangs are valid anywhere in the document, but some can only be used in certain Sections or Modules.

The `!newpage` bang takes no arguments and inserts a pagebreak.

The `!img` bang inserts an image. It takes two arguments: a filepath which specifies the location of the image, and a width of the image. The width of the image should be specified in units that the `\includegraphics` LaTeX command can parse. The image is automatically centered.

The `!gap` bang inserts a vertical space. It takes one argument: a height which should be specified in units that the `\vspace` LaTeX command can parse.

The `!pkg` bang is used to import LaTeX packages. It can only be used in the Header section. It takes any number of arguments; each argument is a name of a package to import.

The `!options` bang specifies formatting options for a Module. The arguments consist of either flags or key-value pairs. The specific arguments that can be set depend on the Module; they are described in the respective Module documentation above.

The `!ans-options` bang specifies formatting options for the answers to a question Module. The arguments consist of either flags or key-value pairs. This is described in more detail later in the documentation.

The `!txt` bang inserts text in a MC Module. It can only be used in a MC Module. It takes one argument, which is all the text that you want to insert. This allows you to insert text in the module without having to start a new Text Module, which would mess up the formatting for the MC questions.

The `!newcol` bang inserts a column break in a MC Module (with the `twocolumn` option enabled). It can only be used in a MC Module. It takes no arguments.

The `!hrule` bang inserts a horizontal line in a MC Module. It can only be used in a MC Module. It takes no arguments.

2.4 Answer Sheets and Keys

Running examtex always produces two .tex files: one for the exam, and one for the answer key. In the case that you also want a separate answer sheet, it will be appended ot the end of the exam (rather than in a separate document). The exact formatting of the answer sheet and key are determined by the `!ans-options` bang in each question Module.

For Matching, TF, and MC questions, you can indicate that there should be an answer sheet by specifying the `sheet` flag in the `!ans-options` bang; the formatting for the answer sheet will be the same as the key (multicolumn grid of blanks). The `break` flag will insert a pagebreak in the answer key/answer sheet at the end of the Module.

For FRQs, it's a bit different. By default, the exam itself will have space in between (sub)questions for the student to write their answers. The `sheet` flag will indicate that no space will be given on the exam paper itself; all answers should be written on the answer sheet at the end of the exam. The `twocolumn` flag will organize the answer spaces in two columns instead of one – this is better suited for questions with shorter answers.

The answer key will always contain the answers only (no questions), organized in the (sub)question hierarchy that the question was written in. The answer sheet will look similar, with more space between questions to write answers, according to the spacing specified (see the documentation above about the FRQ Module). The `break` flag does the same as before.