APEX Formatting Guide

Contents

1 Introduction **1**

2 Writing Exercises **1**

3 An Environment For Creating Examples **3**
3.1 Basic Principles 3
3.2 Options 4

1 Introduction

This manual is intended to guide the user through using a set of macros designed to facilitate the writing of mathematics texts. There are four main sections to this document: writing exercise/answer sections, an environment for writing examples within the text, environments for writing definition/theorem statements (and the like), and a section that highlights some other commands pertinent throughout.

2 Writing Exercises

One could "easily" write exercises for a mathematics text by creating an Enumeration environment and writing each problem as a new `\item`. However, this can be cumbersome for a variety of reasons. Without highlighting the difficulties of this method, we posit the following system obviously takes greater initial effort but with overall positive gain. If one disagrees, feel free to use `\begin{enumerate}`.

A fundamental principle to the APEX format is that each exercise is contained in its own file that consists of two lines. The first line contains the problem statement; the second contains the answer (of course, the content level of the answer is up to the author). Instead of writing problems/answers in a cumbersome fashion to maintain the two line rule, it is much more convenient to enclose the problem and solution each in a set of curly braces, `{ }`. So a sample exercise might consist solely of the following:

```
{Evaluate the following definite integral:
$\int_0^1 x^2\ dx$.}
{$\frac13$}
```

Naming this file properly is important. While the following restrictions may someday be removed, for now it is necessary to name the exercise file with the following convention: *chapnum_secnum*_ex_*probnum*.tex. So the seventeenth problem you've written for chapter 3, section 5 would be named `03_05_ex_17.tex`. Of course, this might not end up being the seventeenth problem of that section, nor must it end up in that section. It is helpful as you write to number your exercises for the section you intend them to end up in; later on, it won't matter what their actual name is.

As one's text grows in size, one will have written *many* exercises, likely over an extended period of time. If one wants to edit the problem in section 4 of chapter 3, one is likely to have forgotten what that actual problem is named. To facilitate finding the name of the problem, one can include the command `\printexercisenames` in the text. All subsequent exercise names will be printed unless the command `\noprintexercisenames` is given. Of course, by default, exercise names are not printed. Make note: printing the names changes the format of the page slightly. Do not try to make final formatting changes to the document while exercise names are shown.

To actually create an exercise section, create a file that contains the names of the exercises to be included in the following fashion. One can name this file in whatever manner they please, but it makes common sense to name is something like `01_05_exercises.tex` for the exercises for chapter 1, section 5.

This exercise file will contain a series of one line commands, all either `\exinput` or `\exsetinput` commands. The `\exsetinput` command will be explained shortly. A typical exercise file will look something like:

```
\exset{01_01_ex_01}
\exset{01_01_ex_02}
\exset{01_01_ex_04}
\exset{01_01_ex_03}
```

The exercise file is effectively read twice in the printing of the document. The first time it is read it is used to create the exercise section. It is read again to produce the answer section. Therefore the `\exset` command pulls double duty, either giving commands to print the question or the answer.

To actually produce the exercise section (likely at the end of the current section), use the command `\printexercises`, as in

`\printexercises{01_01_exercises}`. This command does many things. It first prints the word "Exercises" followed by the current section. All of this is followed by a horizontal rule that extends the width of the text. Then the exercises are printed in two column format, in the `\small` text size. These formatting settings are not too difficult to change; one can delve into the appropriate section of the file `Header_APEX.tex` and make changes.

This setup produces a perfectly serviceable exercise section. However, many times one wants to give students practice by giving multiple problems with the same basic instructions. Instead of writing

```
1. Find the derivative of $f(x) = x^2$.
2. Find the derivative of $f(x) = 2x^3$.
3. Find the derivative of $f(x) = \cos x$.
```

one could just write the instructions once, as in

```
In Exercises 1 - 3, find the derivative of the given function.

1. $f(x) = x^2$.
2. $f(x) = 2x^3$.
3. $f(x) = \cos x$.
```

This type of construction is referred to as an "exercise set." To create an exercise set, one first creates individual problems to be included in the set. This is done in the same fashion as before. Using our example above, problem 1 could be created with the following two lines:

```
{$f(x) = x^2$}
{$f'(x) = 2x$}
```

Note that the actual problem statement does not appear here.

An exercise set file is now created. It consists of the statement of the problem in two parts, followed by a listing of the problems that are to be included in this exercise set.

The macros will automatically determine the numbering to be used in the statement of the problem for the set. The two part statement of the problem consists of the wording before the problem range (enclosed in curly braces), followed by the the wording following the problem range, also enclosed in curly braces. Again, as an example, consider the sample exercise set given above. The corresponding set file would look like

```
{\noindent In Exercises}
{, find the derivative of the given function.}
\exinput{01_01_ex_01}
\exinput{01_01_ex_02}
\exinput{01_01_ex_03}
```

The APEX macros automatically figure out the numbering and place it immediately following the "Exercises" and preceding the " , ". Note that in this example the command `\noindent` is used; that is purely preferential.

In creating exercise set files, care should be taken that no extra lines appear at the end of the file. The problem numbering range is determined by reading the number of lines in the exercise set file; extra lines at the end, even if blank, will give incorrect numbering.

Name this exercise set using the convention used before; the first exercise set one writes in section 7 of chapter 2 could be named `02_07_exset_01`.

To include this exercise set along with all other exercises, in the main exercise file use the command `\exsetinput`, as in `\exsetinput{02_07_exset_01}`.

A complete exercise file that could be used to produce the exercises for a section could look like the following, taken from an actual textbook.

```
\exsetinput{exercises/01_01_exset_01}
\exsetinput{exercises/01_01_exset_02}
\exinput{exercises/01_01_ex_19}
\exinput{exercises/01_01_ex_21}
```

While it is not evident here, there are a total of 16 exercises in this section. The first exercise set contains 10 problems, the second 4, followed by two individual problems.

3 An Environment For Creating Examples

3.1 Basic Principles

Examples are an important part of many textbooks. It is often useful to number these examples so that they can be referenced later on, and certainly one does not want to keep track of such numbering on their own. It is also useful to have a consistent way of signifying that an example is over. There is of course no "right" method of doing this and is subject to personal opinion. The APEX format file gives two ways of accomplishing this that look nice. Each method starts

an example with the word **Example** followed by a number; the example text is matched by a line drawn in the margin. The default method puts the word **Example** in line with the text, while optionally one can put **Example** in the margin to help distinguish it. At the moment, the ability to modify these is restricted by the user's ability to write LaTeX code; simple modifications are likely not simple to implement, and complicated changes are probably ... complicated. A sample example is shown further on in this section.

There are two basic forms of the example environment. The standard, `\example`, takes three arguments. The first is a label for the example that can be used later for reference purposes (using the `\ref` or `\pageref` commands). The second argument is the statement of the problem the example is designed to solve. The third argument is the solution to the problem. The word **Solution** appears indented on the left at the beginning of the solution, placed there by the macro.

An alternate example environment uses the `\example*` command, which takes two arguments. The first is again the label; the second is the complete text of the example. This is used for examples that do not fit the "State a problem then show the solution" model. It does not print the word **Solution**.

If one is pleased with the general output of the default settings, one's final concern is the label used in the first argument. It is the convention of many to use a label such as `ex:01` to denote that this is the first example (whereas the first theorem might be labeled with `thm:01`). However, due to the internal workings of the macro, the colon " `:` " *cannot* be used. It is suggested that the underscore `_` be used instead, as in `ex_01`.

An example is given below.

Example 1 How does one go about creating an example? Show in just a few easy steps.

Solution Creating an example is easy. For instance, this example was created with the `\example` command. The first argument, the label, is `ex_first_example.` The second argument, enclosed in curly braces, starts with {`How does one...`. and ends with ...`easy steps.`} The third argument is the solution, which starts with {`Creating an example`... and ends with ...`self referencing.`} , highlighting the computer scientist's penchant for self referencing.

3.2 Options

There are several options available in adjusting the look of examples. After describing the changes we offer several pages of examples.

In Margin / In Text It was previously mentioned that the APEX format allows one to put the word **Example** either in the text or in the margin. One can switch between the two by using the commands `\exampleintext` and `\exampleinmargin`. They can be used anytime within the text and all subsequent examples will be typeset accordingly until the alternate command is given.

Line Options There are several parameters concerning the lines that are drawn that can be adjusted.

Drawing Lines One can choose to not draw lines. One may be interested in doing this during the proof reading stages of the text. For reasons we'll leave unexplained, it can take as many as three compilations before the lines appear in the correct places. In the

meantime, the lines may appear in odd and distracting places. By default, the lines are drawn. To turn them off, use the command `\nodrawexamplelines` and all subsequent lines will be drawn. To turn them back on, use `\drawexamplelines`.

One can also format the style of the line. It may help to know some Ti*k*Z and `xcolor` commands here. The default format is to draw a `thick` line with color `blue!95!black!30`. This can be changed.

Color One can choose to print in black and white. This is especially good for when a printed copy is to be made, as black and white printed copies of color generally don't turn out quite right. To print in black and white, use the command `\printinblackandwhite`. The default black and white color is a gray-scale color, namely `black!30`. To go back to color, use `\printincolor`.

One may change these colors. To change the color color, use the command `\setcolorlinecolor{`*color*`}`. One may use the standard `xcolor` options, such as `red`, `blue`, `green` or use mixtures as shown above. Use Google to learn more about how to mix the color.

To change the black and white color, use the command `\setbwlinecolor{`*color*`}`. This color doesn't have to be "black and white;" one can make it `green` if you like.

Note: when using either of the above options, not only is that color set, it becomes the default option. So if you set the color color but want to continue printing in black and white, you need to follow with the `printinblackandwhite` command.

Line Style The style of the line can also be determined. To set this, use the `\setlinestyle{`*options*`}` command. The *options* can be any comma-delimited list of Ti*k*Z path options such as `thin`, `ultra thick`, `dashed` or even `->` to draw arrows. See the Ti*k*Z manual for more options. Note: while you are able to include a color within this list, it will likely not take effect. Set the color using the options above.

Help Marks The `\drawexamplemarks` command draws a small circle at the beginning of the example line and at the end of what it considers to be the end of the example. This can be useful in "debugging" the results of an example. Sometimes extra space is added to the end of example, especially if the example ends with the end of an environment. (I.e., you end the example with an enumerated list or some mathematics inside of `$$ ... $$`.) In such cases, a vertical skip of `\baselineskip` is added, which causes the example line to end seemingly far below the actual end of the example. In such a case, it can help to have a small circle drawn to show where LaTeX thinks the example ends. In the above cases, the extra space can be removed using the command `\vskip -\baselineskip`. More/less space can be removed by adding a coefficient in front of the `\baselineskip`.

Sometimes the example ends on a following page, even though all the text ends on the previous page. The culprit is likely and extra `\baselineskip`, but sometimes the trouble is harder to diagnose. Try enlarging the current page with `\enlargethispage\baselineskip` or `\enlargethispage2\baselineskip`, etc., to make the current page larger hence allowing the example to end without starting a new page.

Theorem 1 This is a new theorem

Gregory 1 New definition?

Gregory 2 help

HartmanHartman 1 New box

HartmanHartman 2 Hallow!

See HartmanHartman 2