TEX 102

Simple Documents

Getting started. Open a text editor, such as `pico` or `emacs` on a UNIX system or `edit` or `wordpad` in *Windows*. (In a sense, the simpler the editor, the better; if you use a full-featured word processor, such as *Microsoft Word*, be sure to save your document as "text" or "plain ASCII", not in the program's specialized formats such as `.doc` or `.rtf`.) Type the classic introductory message,

```
Hello, world!
```

Save the file under the name `hello.txt`.

Now bring up a command line (a terminal window in UNIX or a command prompt (formerly known as a DOS window) in *Windows*). Type `tex hello` and see what happens. If TEX is not installed on your system, of course, you will get an error message. If TEX *is* installed on your system, after some brief activity you'll find your prompt sitting after an asterisk: `*` . Type `\bye` (and Return or Enter). You should get a message informing you that two files named `hello.dvi` and `hello.log` have been created.

This experience reminds us that we should have put a command in the `hello.tex` file to end the program. Go back to the editor and type `\bye` at the end of the file (and save it). Now try `tex hello` again. This time the program should run to completion without human intervention.

To examine the results we need a program to view `.dvi` files. On most UNIX systems this program is named `xdvi`; on *Windows* machines it is likely to be called `windvi` or `dviwin` (or whatever the documentation that came with your TEX system calls it). I'll assume that the `.dvi` viewer is command-driven, although mouse-driven ones are also common. At your command line type `xdvi hello` (or whatever the appropriate command is). You should soon see a page image containing the words "Hello, world!" The results are even more clear if you can send the document to the printer. Let's not go over the details of doing that, because (1) they differ from one system to another, and (2) you probably don't want to waste the paper anyway.

If you did print the document, this is what you would see:

1. "Hello, world!" in rather small type near the upper left corner of the page.
2. On detailed examination: the words start exactly one inch below the top of the page and about 1.25 inches from the left side.
3. A page number, "1", near the bottom of the page.

Understanding these results helps us to improve the appearance of the document. First, TEX's default type size is "10 point", but for most documents "12 point" is more readable. So, go back to the `.tex` file in the editor and type

```
\magnification=\magstep1
```

at the top. This tells TEX to magnify everything by a factor 1.2. Second, while we're at it let's put in the command `\nopagenumbers`, since it's not good style to put a page number on a document that has only one page anyway. Finally, the top margin is one inch because in the American academic world, pages ordinarily have one-inch margins, so that's what

TEX gives you unless you tell it otherwise. So why does the left margin seem to be 1.25 inches? The answer is that TEX automatically organizes what you type into paragraphs, and your little paragraph has been *indented* slightly. If you don't want the indentation, type `\noindent` at the beginning of the paragraph. But in the present case, we probably don't want to make a paragraph, but rather something more like a sign. So, let's put the text inside a command like this:

```
\centerline{Hello, world!}
```

Now run the file through TEX and `xdvi` again. Do you approve?

Oh, you want to shout at the world more loudly. A centered line is often a title or heading, so it should be big and bold. Near the top of your file (after the `magnification` command) insert

```
\font\boldtitle=cmbx12
```

Then change the main text line to

```
\centerline{\boldtitle Hello, world!}
```

Satisfied? (We got "bold extended" (i.e., wide) type at the size of 12 points, scaled additionally to 14.4 points by our overall `magnification` command.)

Mathematical text. Now let's type the answer to a homework problem. We can start with the template file `1_2_01p.tex`. At the proper place let's insert

```
 A vector parallel to the line is found by subtracting the
coordinates of the  points:
 $(1,0) - (0,-1) =  (1,1) \equiv \vec u$.
 As $x_0$ we may use any point on the line, say $(1,0)$.
 Now the parametric equation for the line is
$$ \vec x = t \vec u + \vec x_0 = t\pmatrix{1\cr1\cr}
 + \pmatrix{1\cr 0 \cr}.$$
```

Run the file through TEX and your viewer to see the results.

In addition to some of the syntax introduced in TEX 101, this excerpt illustrates several new points:

1. Mathematical symbols to appear in a line of the main text are enclosed in dollar signs. Mathematics intended to appear as a *displayed equation* is enclosed in *double* dollar signs. (If you actually want to print a dollar sign, type `\$` — and similarly for any other of TEX's special characters.)
2. You see here how to type a column matrix. A full matrix with several columns,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

   is typed this way:

   `\pmatrix{a&b\cr c&d\cr}`.

   (The `\cr` means "carriage return"; that will be meaningful to those of you who saw a manual typewriter before they became extinct.)

3. A typical TEX command (control sequence) consists of a backslash followed by one or more letters. TEX knows that the control sequence is finished when it meets either a space or a nonletter (a numeral, punctuation mark, or other special ASCII character). Thus, in the excerpt we had to be careful to put a space after all occurrences of `\vec`, and after `\cr` in the matrix whose elements are letters; but the spaces were optional between `\cr` and `1`, and between `\pmatrix` and `}`, and *before* any backslash.
4. TEX automatically builds whatever you type into paragraphs (unless you tell it otherwise), filling up the lines and even hyphenating words correctly. Therefore, you don't need to worry about line breaks and spacing while you're typing; you hit the Return key whenever you feel like it. It's good practice to put any complicated mathematics on a line by itself in the input file, and to start each sentence or large clause on a new line; these habits make the file easier to edit later. The one big exception is that a *totally blank line* means to *start a new paragraph*. So *don't* put blank lines above or below your displayed equations, unless you really want a paragraph break there; if you ignore this advice, you will get unwanted paragraph indentations and possibly messed-up vertical spacing on the page.

Macros, lists, equation numbers, alignments, and other things. What we typed above was only an answer to part (b) of a homework problem. So let's type `\item{(b)}` before it and go back and add an item (a) (with random content, for the sake of an example):

```
\def\zw{\omega}
 \item{(a)} The Fourier transform of $f(x)$ is defined as
 $$\hat f(\zw) \equiv \int_{-\infty}^\infty e^{i\zw x} f(x) \, dx.
 \eqno (1)$$
 Indeed,
 $$\eqalign{ f(x) &= {1\over 2\pi} \int_{-\infty}^\infty e^{-i\zw x}
 \hat f(\zw) \,d\zw \cr
 &= \hbox{almost the Fourier transform of $\hat f\,$!} \cr} $$
 \smallskip
```

Inspect the results.

This excerpt demonstrates:

1. A simple macro, or definition (substituting an abbreviation for a spelled-out Greek letter).
2. How to number an equation.
3. How to align the expressions in a multiline equation. (Putting *both* an equation number *and* an alignment in the same equation unfortunately requires still a third command syntax, which I'll postpone.)
4. Small horizontal spaces created by backslash-comma.
5. A small vertical space created by `\smallskip`.
6. How to put text inside an equation (and more math inside the text). (Without the `\hbox` you would get "$almosttheFouriertransformof$".)
7. Conversion of paragraphs to itemized lists.

Headers and other special lines. Your class handout says you "**must**" put certain information at the top of your homework paper. So, let's do it. Put this in the `.tex` file somewhere after the initial `\magnification` and `\advance` lines but before the `\centerline` line containing the exercise number:

```
\nopagenumbers
 \line{My Name \hfill {\tt m-name} \hfill  Week 1}
 \medskip
 \line{{\tt http://calclab.math.tamu.edu/\~{}m-name/1\_2\_01.pdf}%
\qquad(revised)}
 \bigskip
```

If you run this file through TEX, you will probably get an error message that there is something wrong with the definition of `\~`. That happens because the template file's designer made an (unwise) redefinition of the tilde. Near the top of the file, place a percent sign at the beginning of the line

```
\def\~>{\longrightarrow}
```

Now everything will work properly (unless that abbreviation for `\longrightarrow` was used in the problem statement).

So, what is new here?

1. The `\line` and `\hfill` commands enable you to overcome TEX's compulsion to reformat everything into a paragraph; they give you some control over where small items appear on the page.
2. The `\tt` command puts its argument into the `typewriter` font used for computer commands, e-mail addresses, and all that. Other available fonts include
   boldface `\bf` *slanted* `\sl` *italic* `\it`
3. The tilde and underscore are special characters to TEX, so they need to be preceded by a backslash when we really just want to print them. The tilde normally goes over the letter that follows it, so we need to put in an *empty group*, `{}`, to play that role.
4. `\qquad` gives a certain amount of horizontal space. (`\quad` gives half as much. A backslash followed by a space gives even less — as much as a typical interword space.) `\smallskip`, `\medskip`, and `\bigskip` give suitable amounts of vertical space.
5. The percent sign `%` turns everything that follows it, up the end of the line, into a *comment*, which is ignored by the program. (Why is a percent needed at the linebreak in our `\line` command? Try leaving it out and see what happens.)

Further information. This presentation is no substitute for a thorough and well-organized compendium of TEX commands, of which many are available if you look.