Avoiding problems, Solving problems, Asking for help

Jonathan Fine
LTS Strategic
Open University
Milton Keynes
J.Fine@open.ac.uk

20 October 2006

Outline

Types of problem

First we discuss four common types of problems.

1. Installation
2. Authoring
3. Compiling (typesetting)
4. Doh! and other Gotchas

Some other types of problems are including graphics, preview, printing and, finally, bugs in TEX the program (very rare).

Here are some recent examples of problems I've dealt with:

- TEX cannot write to output dvi file
- TEX hangs
- TEX creates corrupt dvi file
- Unexplained change in pagination

Earlier run of TEX not finished, network failure, out of disk space, change in math symbol font used in subscripts.

Installation problems

Something very basic is not working.

- TEX the program is identical on all platforms
- Most 'pure' TEX errors are (or should be) cross-platform
- Many preview, printing and graphics problems depend on the platform
- Don't change or update your installation without good reason
- There are distribution-specific newsgroups
  - Mac OS/X: gmane.comp.tex.macosx
  - MikTeX: gmane.comp.tex.miktex
  - TeXLive: gmane.comp.tex.texlive

  Is yours a problem with a particular distribution?
- For each platform, there may be several 'front ends'
  - Windows: WinEdt
  - Windows: Texnic centre
  - Mac OS/X: TeXshop

  Is yours a problem with a particular front end?

Authoring problems 1/2

You know what you want (we hope), but you don't know how to do it in TEX.

- Focus on your thesis/article/book
- Don't write your own complex macros
- Keep close to standard LATEX
- Use the AMS maths packages
- Find a good source for LATEX coding in *your* area. Some examples:
  - *The TEXbook*, by Don Knuth
  - *LATEX user's guide*, by Leslie Lamport
  - *The LATEX companion*, by Frank Mittlebach et al
  - LATEX for logicians website, by Peter Smith
  - *Math into LATEX*, by George Grätzer
  - *LaTeX line by line*, by Antoni Diller

Authoring problems 2/2

Keep it simple. Hegel said:

> *To do something great, one must learn to limit oneself*

- Practice writing simply
- Take care with choice of notation
- Draw only simple diagrams and figures
- How much LATEX do I need to know?
- Learn from the example of your peers (ask your colleagues)...
- ... but you don't have to follow their example
- Use existing symbols whenever possible
- Use existing successful documents as templates
- Use simple typography, e.g. remove rules from tables

Compilation errors

Most of these are TEX errors, such as an undefined control sequence, a missing brace, on an extra brace.

- Use an editor that provides syntax highlighting
- Use a front end that provides templates
- Compile the document often (particularly if you are not so confident)
- Insert { and } as matched pairs
- Insert environment commands as matched pairs
- Copy offending item to top of file, and `\end{document}` early
- Confusion results when you forget to supply arguments to a command
- Sometimes deleting the generated (auxiliary) files clears a problem

We all make errors that are caught in compilation

Doh! and other Gotchas!

There are some things that that LATEX is not well equipped to handle.

1. Verbatim (e.g. typewriter computer code) does not work in macro arguments
2. You cannot wrap verbatim environments
3. LATEX does not provide perfect support for colour
4. The package `hyperref` likes to be loaded last
5. Conversion to HTML is not always reliable
6. Macro packages that make characters active (in addition to ~) often cause problems in LATEX
7. Float placement and complex page makeup can be tricky

Here's an obscure gotcha: `$\alpha\time\beta$` produces α. So where has the β gone?

`\time` is a count parameter, and `\beta` is a `\mathchar`. Author should have written `\times`.

Author defined macros 1/2

This is an interesting and sometimes controversial subject.

- Don't define your own macros . . .
- . . . except for good purposes
- OK: `\newcommand\sigmabar{\overline{\sigma}}`
- Good motives: Save on typing, consistency of notation and typography
- More examples to follow
- Don't change category codes
- Don't try to wrap verbatim environments
- Don't write macros with optional parameters
- Don't load packages just so you can write a macro

Author defined macros 2/2

When you write complex macros, you are ceasing to be an author. Go outside the author area, and there's an awful lot you may need to know.

If you must define your own macros . . .

- Don't redefine existing commands
- Don't invent a new syntax
- Don't change any category codes
- Don't write macros that are 'delimited', e.g. by `\this` e.g. `\def\wibble #1\this #2 { . . .`
- Take great care with stray white space, e.g. after #2 above
- Worth repeating — stray white space

. . . and don't expect help if you post a so-called minimal example that uses these complex macros you defined.

Real life TEX: Green and Tao on primes 1/2

In 2004 Ben Green and Terence Tao proved a major result in mathematics (number theory).

Consider the sequence 199, 409, 619, 829, 1039, 1249, 1459, 1669, 1879, 2089.

Note that

- We have here 10 numbers
- Each number is the previous, plus 210
- Each of these numbers is prime

Tao and Green proved "The primes contain arbitrarily long arithmetic progressions". This is a major result, and helped Tao win the Fields medal this year (roughly equivalent to Nobel prize).

They wrote it, of course, in LATEX.

Question: How much LATEX did Green and Tao need to write this paper?

Real life TEX: Green and Tao on primes 2/2

LATEX source downloaded from http://www.arXiv.org, and used here with authors' permission. (See Appendix to handouts.)

1. Only standard packages used
2. Hand fiddling of dimensions
3. Simple numbering scheme for theorem-like elements
4. About 30 author's convenience macros
5. Commands defined using `\def` (TEX primitive) instead of `\newcommand` (tut, tut)
6. Some of the definitions could be improved

Conclusion: You don't have to be a LATEX wizard to be successful (and win a major prize) in mathematics. (And you can download LATEX source for papers in your field.)

Real life TEX: Printing handouts from beamer 1/2

1. Search internet to find pdf for beamer manual
2. Print out documentation (214 pages!) to protect my eyes
3. A piece of luck. Table of Contents: Creating Handouts
4. Beamer manual says use the command
   `\pgfpagelayout{2 on 1}{a4paper} % from beamer manual`
5. Get and read the log file error message

```
(C:\texmf\tex\latex\beamer\themes\outer\beamerouterthemec
(C:\texmf\tex\latex\pgf\utilities\pgfpages.sty
(C:\texmf\tex\latex\tools\calc.sty))
! Undefined control sequence.
l.4 \pgfpagelayout
                  {2 on 1}[a4paper] % from beamer manual
?
```

6. Open the file
   `C:\texmf\tex\latex\pgf\utilities\pgfpages.sty`

Real life TEX: Printing handouts from beamer 2/2

1. From last slide, open the file
   `C:\texmf\tex\latex\pgf\utilities\pgfpages.sty`
2. Re-read the log file error message

```
! Undefined control sequence.
l.4 \pgfpagelayout
                  {2 on 1}[a4paper] % from beamer manual
?
```

3. Search for ‘layout’ in `pgfpages.sty`
4. Find the following, and follow its advice

```
% Use a layout
% #1 = layout name
% #2 = options
% Example:
% \pgfpagesuselayout{resize to}[a4paper]
```

5. Be a careful good citizen, if you can (Soureceforge bug id 1490542)

Problem solving tools

- Syntax highlighting in your editor
- Version control (commit successes and problems)
- The `log` file
- To skip (include) text, `\iffalse` (`\iftrue`) and `\fi`
- (Advanced alternative, `\ifcase 1` and `\ifcase 0`)
- `\tracingall` (and expect a large log file)
- `\tracingpages = 1` (and expect a large log file)
- The `dvitype` program
- The `cmp` and `diff` programs
- For font problems, `tex testfont` to make font table
- For graphics problems, include graphics only test file
- Preamble only test file, as template for minimal example

Searching comp.text.tex and the internet

1. Search internet and newsgroups as appropriate
2. I find the internet (web pages) better for problems related to a specific program (such as dvips)
3. I find comp.text.tex better for TeX macro problems
4. Use error messages as search keys
   e.g. `! LaTeX Error: File ‘pgfcore.sty’ not found.`
5. You don’t need to understand an error message to copy and paste it
6. You may need to exercise some ingenuity in your search
7. Think carefully about how you would describe the problem
8. Use key words from the description of the problem
9. Take care *not* to include irrelevant words in your search - e.g. if a TeX macro problem, don’t include MikTeX

Creating a miminal example

If you present a complicated problem to solve to the experts on comp.text.tex, and there is nothing clearly wrong with your input, they will ask for *a minimal example*. A small one will be enough though.

How to create a minimal example:

- This is an art, rather than a science. However, the binary search (or 'divide-and-conquer') is a useful tactic.
- Getting the minimal example can be hard, e.g. when page breaks are unexpected.
- Do look at TEX's log file. It contains many clues.
- Read error messages from TEX very carefully — TEX does not know what is supposed to be happening, but it's always right about what is happening.
- I find using `\tracingall` and then jumping to the failure in the log file very useful.

Once you get to a minimal (or small) example, the problem is already half-solved. (But half-proved is not-proved.)

Asking for help

If you are asking someone in person, they may appreciate your providing them with *all* the relevant information. What to do will depend on your relationship with the person, and their preferences.

If you are asking for help on a newsgroup or email discussion forum:

- Try to choose the appropriate forum. What sort of problem do you have?
- Have you searched for the answer yourself?
- Have you produced a minimal example?
- Choose a good title. Here are some recent poor titles (with better title in parentheses):
  - Metafont (dvips does not find home-made Metafont font)
  - dvips (Error: dvips: ! out of string space in dospecial)
  - Tiff image (Can I include TIF images?)
- Thank people for their help and efforts

A minimal \tracingrestores example

When debugging some LaTeX macros, I wanted to know what values were restored (global assignment) at the end of the group. What output do you expect from this input?

```
\immediate\write16{Wish to see end of group restores}
\begingroup
{\let \aaa \relax} % don't want to see this
\let \xxx \relax   % do want to see this
\tracingrestores = 1
\let \yyy \relax   % and want to see this
\endgroup
```

This is what TeX produces

```
Wish to see end of group restores
{restoring \yyy=undefined}
```

Why don't we see the restoration of `\xxx`? Is this a bug in TeX?