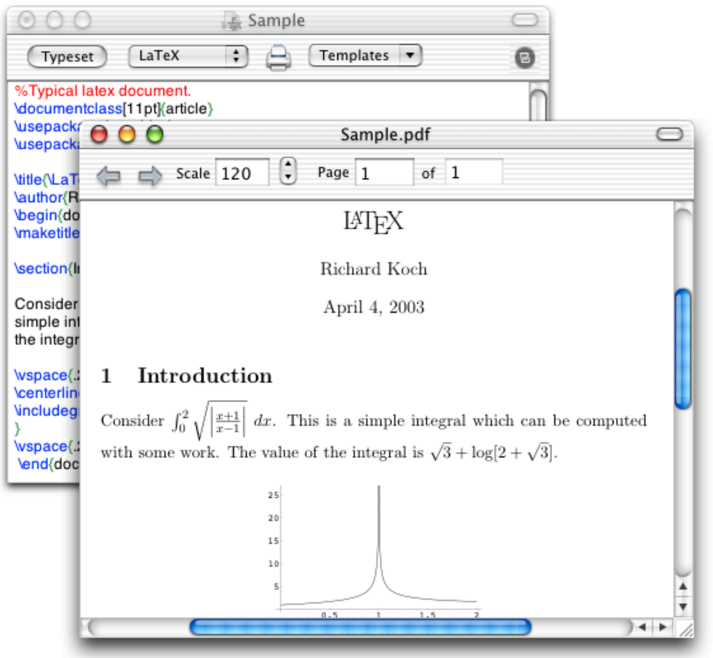
TeXShop in 2003

Richard Koch
Mathematics Department
University of Oregon
Eugene, Oregon, USA
koch@math.uoregon.edu
http://www.uoregon.edu/~koch

Abstract

TeXShop is a free TEX previewer for Mac OS X, released under the GPL. A great many people have contributed code to the project. TeXShop uses teTEX and TEX Live as an engine, typesetting primarily with pdftex and pdflatex. The pdf output is displayed using Apple's internal pdf display code. I'll discuss recent changes in the program and plans for the future.



```
%Typical latex document.
\documentclass[11pt]{article}
\usepack
\usepack

\title{\LaT
\author{R
\begin{do
\maketitle

\section{I

Consider
simple int
the integr

\vspace{.
\centerlin
\includeg
}
\vspace{.
 \end{doc
```

LATEX

Richard Koch

April 4, 2003

1 Introduction

Consider $\int_0^2 \sqrt{\left|\frac{x+1}{x-1}\right|}\,dx$. This is a simple integral which can be computed with some work. The value of the integral is $\sqrt{3} + \log[2 + \sqrt{3}]$.

Introduction

I talked about TeXShop at the 2001 TUG conference. Mac OS X had just been released and still had quirks; during my talk, the Finder crashed. I typed option-shift-escape to bring up the Force Quit panel, restarted the Finder, and continued. After the talk, an audience member told me "I'm not very interested in your program. But it's wonderful that you could restart the Finder without rebooting!"

A lot has changed since then. The Finder never crashes, the teTEX/TEX Live distribution is stronger, lots of pdf bugs have been fixed, and TeXShop has new features.

What are we talking about here?

In case you don't work on a Mac or haven't used the program, I'll start with an overview. When TeXShop first starts, a single window opens for the source code. This window is shown behind another window. The "Templates" button on the toolbar is a pulldown menu naming various pieces of text which can be added to the document; one of these pieces inserts starting code for a LATEX document. The default LATEX template text is shown in blue. Actually the editor colors all TEX commands blue and these have come from the template. A beginner would choose this template and add text in the middle.

Pushing the "Typeset" button on the toolbar causes the document to be saved and typeset with pdflatex. A second window then appears showing the output. This window has tools to change pages, change the magnification, and rapidly move to a later page.

Next to the "Typeset" button is a pulldown menu labeled "LaTeX". This menu contains a list of possible typesetting commands: Plain TEX, LATEX, BibTEX, ConTEXt. Also included are MakeIndex, MetaPost, and MetaFont. If one of these is chosen, the Typeset command will run that program.

I have described the simplest situation. Most users will open an existing TEX document and then both the source and preview windows will open side by side. After adding text to the source, users push "Typeset" to save and typeset the changed source. Multiple documents can be open at once.

Printing is straightforward. Select "Print" and the output will be printed on any inkjet or PostScript printer. If the printer handles color, illustrations and text can appear in color.

The Mac OS X landscape

There have been many changes in the Mac world since 2001. The most important from my point of view is that bugs in Apple's pdf code have been quashed. I know of no outstanding bug.

My bug folder lists the following items, in order of correction:

- The Window Server Crash Bug. After one early release of OS X, users sent me illustrations and complained that TeXShop crashed when it tried to display them. When I saved these illustrations in a folder and selected one, the Finder tried to create a thumbnail preview and crashed. Apple fixed this bug very rapidly!
- The Accent Bug. Certain accented letters displayed with the accent displaced to the right rather than placed over the letter. Documents with accents printed fine on PostScript printers, but incorrectly on inkjet printers. This bug was fixed gradually, and as a result complaints from European nations gradually narrowed to just a few countries, and then disappeared.
- The Ghostscript 7 Bug. TeXShop has a second typesetting option in which a source file is typeset with LATEX, the dvi file is converted to PostScript with dvips, the PostScript file is converted to pdf by Ghostscript, and this pdf file is displayed. For a long time, Apple's pdf routines ignored most fonts in documents created by Ghostscript 7 and the user ended up with a page sparsely sprinkled with letters. Ghostscript 6 worked fine. This was a serious bug because NSF preprint repositories often had Ghostscript 7 pdf files. The bug was fixed in Jaguar and TeXShop users now use Ghostscript 8.
- The Large Letter Bug. Large symbols — integral signs, summation signs, parentheses — were cut off. Sometimes the top and bottom would display and the middle would be missing. Documents with such symbols would print fine on PostScript printers, but not on inkjet printers. This important bug was fixed in system 10.2.4.

Another significant change is Gerben Wierda's rapid improvement of the teTEX/TEX Live installer, making it easy for users to maintain an up-to-date TEX distribution. Wierda's i-Installer is completely independent of TeXShop and his distribution is used by most front ends on Mac OS X. I often receive email complaining that TeXShop doesn't typeset correctly. If the email contains an example, I typeset it from the terminal and usually determine that the typeset output is exactly as the writer describes. So I write back "this is not my problem." If I were more responsible, I'd look at the source code and try to find the error.

A final change is that TEX can now be created on Mac OS X using a great many different programs. Apple's release of its own X window system makes X programs coexist nicely with Aqua programs, so users can easily use emacs and xdvi. At the Apple developer conference, I only saw one slide of a system running TEX and that fellow was using xdvi. Many wonderful front ends and tools have been released; these will be described by other people at the conference. I'm somewhat selfishly going to describe only TeXShop, but users should experiment before settling down with one solution.

Configuring TeXShop

In the course of this talk, I'm going to mention several people who have made enormous contributions. Don Knuth, of course, is responsible for everything we do; I've never met him. Two other people are here, Thomas Esser who created the teTEX distribution, and Hàn Thế Thành, who created pdftex. Salute!

TeXShop now has a reasonable help system, created by Martin Kerz (figure 1). I've haven't been a great fan of Apple's Help Viewer because it is slow, but I expect great improvements soon. Let me show you the help system because I want to point out one

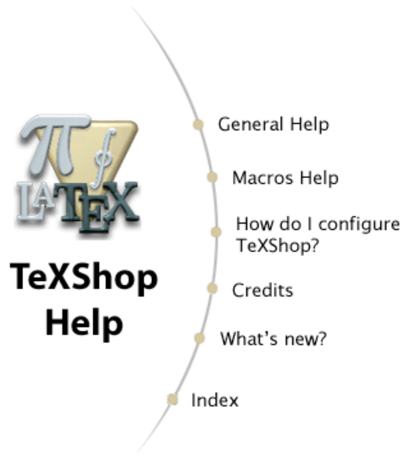
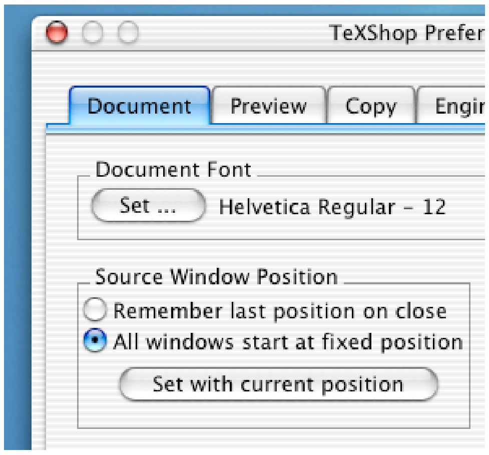


Figure 1: TeXShop help system.



Document Font: Helvetica Regular – 12

Source Window Position
- Remember last position on close
- [✓ All windows start at fixed position]

Set with current position

Figure 2: TeXShop window positioning.

feature, the "How do I configure TeXShop?" section. I hope users will get in the habit of reading this section whenever they upgrade the program.

In the "Recommended Preference Changes" section, I recommend different window placement defaults. When TeXShop is first installed, it is configured to remember the last positions of the source and preview windows, and use these the next time TeXShop runs. This only makes sense if you work with one file at a time; otherwise it is likely that one of the windows was pulled out of the way before quitting, and its position may become the position TeXShop uses when it runs again. It is far more convenient to open source and preview windows side-by-side with fixed dimensions set in advance. To make this happen, typeset a small file and position the source and preview windows as you like. Then open Preferences. For "Source Window Position," choose "All windows start at fixed position" and push the "Set with current position" button. Configure the Preview window similarly (figure 2).

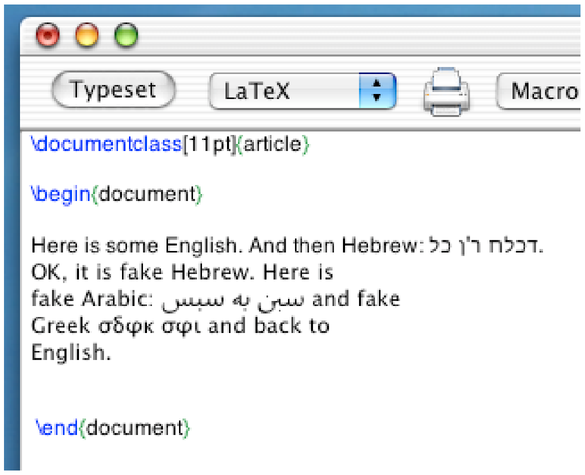
While I'm dealing with preferences, I'd like to mention another one. TeXShop now starts configured to automatically convert eps files to pdf and tiff files to png during typesetting. We made this change when pdftex dropped support for tiff files. In the new configuration, the `-shell-escape` flag is set for TEX; the flag gives TEX permission to run other programs during typesetting. The intended other program converts graphic files, but of course it could be any program. A few of our users worried that malicious code could be distributed as TEX source; I suspect that those users were college teachers with dangerous students. One fix is to remove the shell-escape flag, which preferences permits. But there is also a new "Shell Escape Warning" check box in preferences. If it is set, then *the first time* a document is typeset during a TeXShop session, a warning dialog appears allowing the user to turn off shell escape for that particular document. This makes it easy to use graphic conversions for your own source but not for source received in email. In case you are wondering, I don't have the flag checked on my machine. Live dangerously.

TeXShop has a few hidden preferences with no interface in the preference panel. One causes it to create a backup file every time a file is saved or typeset. The configure help section explains how to set that preference.

File encodings and the power of programmers in Japan

Internally, Cocoa editors use Unicode. But conventional TEX cannot handle Unicode, so TeXShop converts a file to Unicode when it opens the file, and converts the Unicode back to something else when it saves. TeXShop now supports sixteen different encodings; I don't recall how many encodings TeXShop supported in 2001 — maybe only one.

Three of these encodings are interesting. The first is utf-8, which preserves Unicode. I know a little about TEX programs which can input Unicode, but hope to learn a lot more at the conference. To enter Unicode text, open System Preferences, choose the International module, and select the Input Menu tab. Add extra languages by checking the boxes on the left. A small flag will appear on your menu bar. To type in English, select the US flag (I suppose other flags also work); select other flags to input other characters. Notice that both Arabic and Hebrew are inserted right to left, and notice that ligatures work in Arabic, a language in which letters change shape at the end of a word. (See figure 3.)



```
\documentclass[11pt]{article}

\begin{document}

Here is some English. And then Hebrew: דכלח ר'ן כל.
OK, it is fake Hebrew. Here is
fake Arabic: سبن به سبس and fake
Greek σδφκ σφι and back to
English.


\end{document}
```

Figure 3: Unicode input.

Utf-8 encoding is slightly tricky because not all byte strings form legal input. Users sometimes open a document in utf-8 which they created in another encoding and just get a blank window because the conversion routine gave up. TeXShop never writes to files when they are opened, so the edit window should just be closed and the file opened with a different encoding. But in a panic, some users try to typeset the blank window. This is not a good move because TeXShop writes before typesetting. Users who often change file encoding should set the hidden preference to make backup files as protection against this sort of error.

A few months ago, a user asked for Russian encodings. Further email revealed that there are lots of Russian encodings in use; TeXShop now supports five such encodings. Users must deal with more than one encoding because they receive email. So in version 1.30, TeXShop supports selecting the encoding when a file is opened and when it is saved. When opening a file, the default encoding will be the one chosen in preferences, so most users can ignore the entire matter. If the encoding is changed, then the default for that one file when it is saved will be the new encoding.

Students of mine from Japan sometimes report that their friends have trouble with TeXShop. It was difficult to get a clear explanation because my students weren't TEX users themselves. A year after learning of the problem, I got a wonderful email from Seiji Zenitani with a clear explanation *and* the code to make TeXShop work. If I understand correctly, in System 9 Japanese fonts displayed the backslash character as a Yen symbol, and the Japanese keyboard had a yen symbol on the key which produced a backslash character. This created no problems for TEX, of course. But when Unicode was introduced, it became necessary to separate the two symbols, and the Japanese keyboard began emitting a real yen symbol. The Japanese coped with this change by modifying TEX to use yen instead of backslash as the TEX control character. I mentioned this to a colleague with collaborators in Japan, and he said "yes, the damn yen problem."

At any rate, Zenitani took care of the problem and also localized TeXShop in Japanese. It is the localization I like best! Then around the time version 1.27 was to appear, Zenitani wrote me that he had heard of some smoother routines by someone else in Japan, who would write me soon. In that way I made contact with Mitsuhiro Shishikura, the most powerful programmer I know. Shishikura is a mathematician. He just told me that he has become chair of his department and I sent condolences.

Shishikura sent code to simplify TeXShop's handling of the yen problem. Then he casually mentioned that he had added a magnifying glass for the preview window (this is listed in my 2001 article as an important missing feature). So TeXShop finally had a magnifying glass. I was going to show you the glass below, but screen capture doesn't show it; apparently Shishikura's code is as mysterious to the Mac as it is to me.

No sooner was Shishikura's magnifying code in the program than he invented a Macro editor as well. I'd like to show you one nice feature of that editor. TeXShop has hidden preferences to set the background color of the source window and of the preview window, but these preferences must be set in Terminal. Suppose you wish to experiment with various backgrounds in preview documents. Create an Applescript macro in the Macro editor using the following code.

```
--AppleScript
do shell script "defaults write
          TeXShop Pdfbackground_R 0.5"
do shell script "defaults write
          TeXShop Pdfbackground_G 0.5"
do shell script "defaults write
          TeXShop Pdfbackground_B 0.5"
```

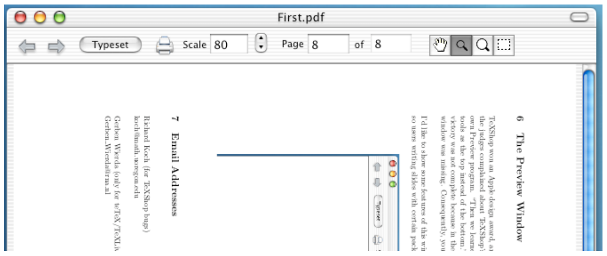
After executing this command, the background of the preview window will be gray the next time you typeset. Don't get too excited; the background will not print, but is useful for previewing if your document has colored text which is difficult to see against white.

The TeXShop source code has a folder containing the email Shishikura wrote when he sent these modifications. It is a model of describing proposed changes. Shishikura listed the changes needed in the nib files, and in each Objective C file. He described each new feature separately, so I could include some and not others. This contrasts with code changes I receive occasionally that are so massive they cannot be digested. Every source file in the project has been modified and as far as I can tell, there isn't a scrap of remaining code which I wrote myself. Moreover, the interface is completely different. What am I supposed to do with such a suggestion?

The Preview window

TeXShop won an Apple design award, and after the ceremony one of Apple's engineers told me that the judges complained about TeXShop's pdf window, whose design does not follow the design of Apple's own Preview program. "Then we learned," he said, "that Preview was redesigned in Jaguar with tools at the top instead of the bottom." Apparently that placated the judges. But our victory was incomplete — in the poster Apple put up after the ceremony, the TeXShop pdf window was missing! You couldn't tell from the poster what TeXShop actually did.

I'd like to show some features of the preview window introduced since 2001. First, the view can be rotated so users writing slides with certain packages view them in correct position.



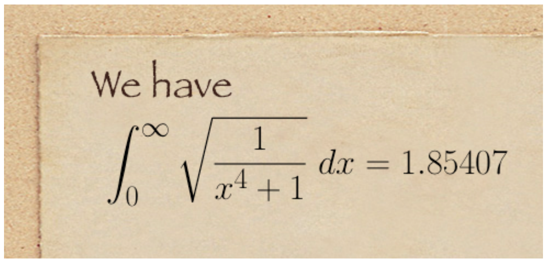
I have often taught linear algebra. After writing the code to transform window coordinates and then testing it and discovering that the window was transformed God-knows-where, I have new humility. Any linear algebra course I give in the future will be error forgiving.

For version 1.30, just released, Shishikura revised the pdf display code, introducing new display options like scrolling through the entire document and displaying two pages side by side. He cleaned up the code tremendously, so pages are centered rather than displayed in the bottom right of the window and ... well, I don't want to think about the old code.

Indeed, there are so many new options that a couple have minor bugs which will be cleaned up soon. The lesson is that if I always use "Single page" option and Shishikura always uses the "multipage" or "double multipage" option, then nobody is testing the "double page" option.

Copy and paste; drag and drop

Shishikura also added the ability to select a portion of the pdf output and copy it to the clipboard or drag it to another application or the desktop. This procedure can be controlled by new preferences or by menu commands. For example, the file type of the copy can be pdf, tiff, jpg, png, or pict; the foreground and background colors of the copied image can be selected in preferences for some of these types.


We have

$$\int_0^\infty \sqrt{\frac{1}{x^4+1}}\,dx = 1.85407$$

Above, for example, is a section of text copied into a Keynote slide in pdf format. But users working with Keynote should also consider J. McKenzie Alexander's wonderful *Equation Editor*. See the site evolve.lse.ac.uk/software/EquationEditor/.

The LaTeX panel

Geoffroy Lenglin created a great symbol panel for TeXShop. Clicking on a symbol inserts the appropriate source code into the document.

I'd like to tell a little of the story of this panel because it shows that a contributor should not interpret silence as a "no". Lenglin sent this panel while I was working on other TeXShop features, so it didn't immediately appear. I couldn't just glue it in because there were interface features to consider: hiding the panel when the preview display is active, remembering the Panel position when TeXShop quits, etc. Then tasks at the University intervened and Lenglin heard nothing from me for many months. When I finally had time, I wrote Lenglin something like "you may think I've forgotten, but now I'm ready to add the LaTeX Panel." He replied "it is good that you wrote, because I have just finished a masters degree in Aeronautical Engineering at MIT and this email address will become inactive. I'm leaving for France." Whew!

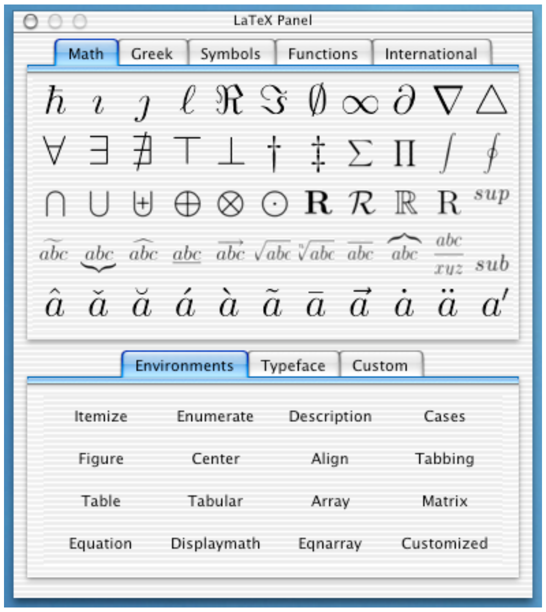


Figure 4: LaTeX symbol panel.

The above quotations are approximate because I can no longer find those messages in Mail. But I discovered old email from him containing a French localization of the help files. They didn't get into TeXShop! Apologies. If you write me and I ignore the email, please write again. And again.

Split window editing, syntax coloring, etc.

There have been other changes. The editing window can be split so you can view one section of source while writing another. Various methods have been added to speed typing; for instance, you can request that pressing the double quotation key insert the TeX code for " " with the cursor placed between the quotation marks. Command completion is also possible; typing part of a word and pushing escape will cycle through all possible completions, and the completion can be a complicated phrase which need not contain the letters originally typed. (However, this system does not use an existing dictionary, so words must be added before they are available for command completion. Improvements to this system will appear in the future.)

For many months, users complained that when all windows in TeXShop are closed and another application is temporarily active, clicking on TeXShop in the dock created a new empty source window. My response was that the Apple Interface Guidelines require this behavior. Indeed the behavior is automatic in Cocoa. Eventually, Gerben Wierda stumbled across the "applicationShouldOpenUntitledFile" call in the API's and he wrote: "maybe the Guidelines require it, but Apple themselves provided a way to get around the guidelines." So a preference item now allows users to control this behavior.

When Jaguar appeared, TeXShop syntax coloring slowed way down. Users who let the program wrap lines and thus create very long paragraphs had real problems. I spend half of 2002 working on this problem ... don't remind me.

Other changes have been made. The "root file" design was improved following a user suggestion, Zenitani fixed the tags menu so several tags can have the same text, TeXShop can display ps, dvi, jpg, tiff, pdf, and eps files, converting to pdf when appropriate. The version history file can be consulted for a full list of changes.

The TeXShop design philosophy

I want to tell you about one of my colleagues. His office is two blocks from mine, and he sometimes calls with computer questions. A typical session goes like this:

```
I just received email and it contains
tex code. How do I typeset?

Is the code in an attachment?

No. There's a begin[document] and
then lots of English.

Ah. Copy the source, paste it into a
blank TeXShop window, and hit the
Typeset key.

Wait a minute. Let me get a piece of
paper. OK. Step one. Copy the source...
```

My colleague learned TeX and he started writing books. Wonderful books. Mathematicians read his books even when they are in a different field.

You have a natural question. "He started learning TeX? How many support calls did *that* require?"

The answer is *none*. My colleague immediately understood the connection between TeX and mathematics; he got a few books, and he proceeded. As for software, he doesn't want it to interfere with his life. Once he learns how to do something, he doesn't want to learn a faster way, or upgrade and get more features, or find more buttons on the interface. I went to his office and discovered that he used the original TeXShop preferences and his windows opened at random spots. "I know a better way," I said, "let me change that." "No, no. . . "

I write TeXShop for colleagues like that. The highest compliment you can give is "this program is pretty simple." I'm happy to add features for advanced users, but only if they don't cause beginners to say "what's this button for?" TeXShop should live in the background. It should be there if it is needed but it shouldn't intrude.

The future

When users send suggestions, there is a temptation to pick those that are easy to do, add lots of minor features, and ignore the central issues. Let me concentrate now on two primary requests. I don't know how to do either one, so no promises are being made.

The first request is for an editing option to provide hard line feeds when the editor wraps text. *Textures* works like that; as a user types, the lines wrap; if the window is later resized, the line feeds remain and the lines do not reformat to fit the new window size. Users who request this feature complain that when they send TeXShop files to friends, the friends have versions of TEX which cannot deal with long lines. They also say that some utilities like diff have trouble with long lines.

People have different typing styles. I am in the habit of typing a line feed before I come to the end of a line, so I'm never bothered by these problems. When users send me files with long, long paragraphs, I'm surprised. Hard line feeds go against the basic philosophy of TeXShop because they do something behind the user's back. And hard line feeds are certainly not required by the standard programs in teTEX/TEX Live.

Of course, it is easy to write a three line C program which can convert a TEX source file to a file whose lines are all short. I've sent such programs to people who've written me.

Cocoa doesn't readily yield up the soft line feeds is inserts when it formats for the screen. I've read through the text API's and I don't see an easy way to proceed. But a few users are very insistent that they need those line feeds!

I've tried to postpone until the bitter end the main request by far: synchronicity. With this *Textures* feature, a user can click on a spot in the preview and immediately be taken to the corresponding spot in the source code; conversely the user can click on a letter in the source code and be taken to that letter in the typeset preview.

This feature has been implemented by other TEX preview systems, so in the last few weeks I've been reading about them. I've read about source specials in dvi files, about the src-special flag for tex and pdftex, about dv2dt, about dvii, vpe, and srcltx. These things look helpful for a rough version of *Textures* synchronization, but I need help. Big time help.

Very early in the life of TeXShop, a user asked for synchronization using approximately the following words: "and by synchronization I do not mean that I'll be happy if it takes me to the same paragraph; I insist to be taken to the *exact* corresponding character in the source! That or nothing." Gosh! Intimidating. Blue Sky doesn't charge money for nothing.

TeXShop has two typesetting modes. One produces a dvi file (and uses it to create a pdf); the other produces a pdf directly. It is possible that some of the utilities listed above would help with one method but not the other. If I must choose, I'd much rather have synchronicity using pdftex than synchronicity using tex with dvi files.

Perhaps one of you has a complete solution ready to go; it pays to ask.[1] I'd really like to see a flag in pdftex which causes it to output a separate file containing this information:

1. the name of the source file; when it changes due to input statements, the name of the new source
2. for each individual character typeset, the line number and position within that line of the character in the source, and the page number and location in pixels of that character after it is typeset.

I wouldn't know how to handle sections of text created by macros.

Maybe that is way too much. I'm going to end with a list of questions.

- Are there utilities on the web which can decipher a dvi file and print a readable list of the contents, or of part of the contents?
- For example, could such a utility print the location of the start of each line, and the position within that line of the start of each character?
- Are there utilities on the web which can decipher a pdf file and print a readable list of the contents, or of part of the contents?
- Adobe Acrobat allows users to drag the mouse and select a portion of text. It allows users to search a pdf document. How are those tasks done?
- pdftex has a flag named `-src-specials`. Is there a utility which can read a pdf file and output these source specials?

If you have any answers or information about these matters, please talk to me during the conference or write afterwards.

[1] Since this paper was written, `pdfsync.sty` has been achieved through the efforts of many volunteers. See Jérôme Laurens' forthcoming presentation on this at TUG 2004. *Ed.*