Dotex — Integrating TEX into the X Window System

Anthony J. Starks
Merck Research Laboratories
Computer Resources, RY86-200
126 Lincoln Avenue
Rahway, NJ 07065 U.S.A.
Phone: (908) 594 – 7288
FAX: (908) 594 – 1455
Internet: ajs@msdrl.com

Abstract

Dotex is a system to integrate TEX tools and automate the format, edit, preview, and print cycle in the X Window System. Dotex uses a single X Window client, xtmenu, to provide a simple push-button interface to the TEX formatter, text editor, and dvi previewer. Other functions such as spell-checking are easily added. Dotex's function is to integrate, not change existing tools, providing a highly interactive "point and shoot" interface to traditional batch-oriented programs. Dotex's chief advantage is allowing the user to rapidly visualize changes in the manuscript, thus facilitating such things as prototyping different typographic effects.

Introduction

TEX is an interactive, terminal based program, but today's computing environment is increasingly window- and mouse-based. Dotex provides a "point and click" wrapper around TEX and other tools in the X Window System (Gettys et al, 1990).

Dotex was built out of the frustration of constantly running an editor and the TEX formatter during the document preparation cycle. The first step was to use the built-in history and line editing functions of the shell command interpreter, but even this was unsatisfactory. What was needed was a tool that would centralize all the actions needed to prepare a document, without having to change the tools themselves. Out of this need, dotex was born.

The Tools

The tools needed to implement dotex are the front-end client, xtmenu, the dvi previewer xdvi and a text editor. The standard X terminal client, xterm is used to start dotex and serves as a logging device.

xtmenu. The X client, xtmenu is the heart of dotex. Xtmenu's function is to bind actions to buttons. When a button (or keyboard equivalent) is pressed, a user-defined action takes place. The action can be some window-management task, or more importantly, the execution of an arbitrary program.

xdvi. xdvi is a typical dvi previewer that has several attractive features:

- It uses the same font bitmaps as the printer, so that a special set of font bitmaps is not needed just for the previewer.
- xdvi has a rich, unobtrusive interaction model for moving around within the dvi file, including jumping to a particular page and zooming to different magnifications. A pop-up magnifier for examining fine typographic details is also provided.
- Also, recent patches provide much improved display quality through anti-aliased font display on color or grayscale X servers.

But the critical feature needed for the success of dotex is xdvi's ability to refresh its display when the dvi file changes.

The editor. Any standard text editor may be used with dotex. Popular editors usually found in a X Window environment such as Emacs and *vi* will work fine. The editor must be able to write its buffer without quitting to be effective with dotex. Window based editors such as xedit are particularly effective because they don't need to be run within an xterm, and already support mouse-based interaction. The author's personal favorite is mx, a programmable mouse-based editor based on the Tcl language (Ousterhout, 1990). Mx provides mouse-based selection, multiple windows, and other niceties such as support for regular expressions. Another welcome feature for TEX use is the visual indication of the nesting level of delimiter pairs like { } and [ ].

Other tools. Any program that can aid in document preparation can potentially be used with dotex. For example, spell checking can be added by running an interactive speller such as ispell. In the same way, TEX manuscripts may be checked for grammatical correctness.

Configuration

The dotex shell script. A simple xtmenu script looks like this:

```
 1. #
 2. #
 3. #  dotex -- automate TeX and tools
 4. #
 5. #  Usage: dotex file
 6. #
 7. file=$1
 8. xtmenu -noquit -stdin <<!
 9. "$file.tex" !label
10. TeX         %tex $file.tex
11. Edit        %mx $file.tex&
12. Preview     %xdvi $file.dvi&
13. Print       %dvidsk $file|lp -o nobanner -r
14. Done        !exit
15. !
```

Figure 1: The dotex script

It is a UNIX shell script that defines the buttons in the main xtmenu window. Line 7 saves the name of the file that is to be used later in the script. Line 8 is the execution of the xtmenu program itself. The option -noquit tells xtmenu not to quit after an action executed, and the -stdin option means take the input from the standard input file; in this case the shell *here document* (Kernighan and Pike, page 94) contained in lines 9 – 14. If desired, the buttons may be arranged horizontally by adding the -horizontal option.

The lines in the *here document* are the heart of dotex. Each line is a label/action pair. The first string in the pair is a label that manifests itself as either a button or label in the xtmenu menu. The second string is either a keyword defining a label, a system command, or special action. The label/action pair in line 9 defines a label at the top of the menu. This label is for identification only and no action takes place when it is clicked with the mouse. Lines whose action-string begin with % are sent to the command interpreter for execution when their labels are "pressed" by placing the mouse cursor over them and pressing the first mouse button.

Lines 10 – 13 run TEX, the editor, the previewer, and dvi print tool respectively on the target file. The action in line 14 terminates a dotex session, and line 15 terminates the *here document.*

The order of the buttons is arbitrary, but it is best to arrange the most used buttons at the top, and to group the functions logically.

Note that the definitions of the buttons are not necessarily static. With appropriate X resource settings, a given key sequence may be defined which invokes a dialog box in which a new action and/or label can be defined. This mechanism can be used to alter command names or parameters.

A dotex Session

To begin a dotex session, type the command:

```
dotex file
```

in a xterm window running a shell. To make effective use of screen space, the shell window should be about 10 lines long.

This creates the dotex menu, with its buttons corresponding to the script described above:

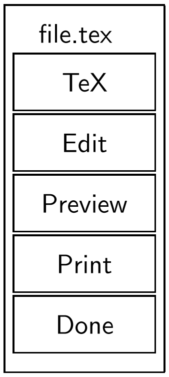


Figure 2: The dotex menu

Generally a dotex session will have these windows on the screen concurrently:

- the dotex menu,
- the editor window,
- the previewer window, and
- the shell, or log window.

The editor and preview windows correspond with the Edit and Preview buttons. The shell window is the xterm used to start dotex. The standard TEX dialog and other system messages are displayed here. If the operating system supports it, `dotex` can be placed in the background*, and then other commands may be issued in the shell window.

Editing. Pressing the `Edit` button creates an editor window acting on the filename specified in the `dotex` command line with `.tex` appended to it. This is usually the first action. When the file is ready to be formatted by TEX, the edit buffer is written.

Formatting. The next action is running TEX on the file just written by simply pressing the `TeX` button on the `dotex` menu. The normal TEX dialog appears in the shell window.

Note that it is best to position the editor window close to the `dotex` menu so that the operation of writing the file and formatting requires minimal mouse movement.

To achieve tighter integration of the edit-TEX cycle, the act of writing the editor buffer could trigger automatic formatting by making the action of the `TeX` button execute this shell script:

```
touch $1.tex
while :
do
    if newer $1.tex $1.dvi
    then
        tex $1.tex
    fi
    sleep 5
done
```

Figure 3: Triggered TEX script

which automatically runs TEX when the source file is newer than the corresponding `dvi` file. The responsiveness of the formatting is controlled by changing the number of seconds in the `sleep` command. As an extra bonus, the file would also be formatted after any other action that accessed the TEX source such as spell checking.

Previewing. If the TEX run produced no errors, you can preview the file by pressing the `Preview` button. This will bring up a window representing the output of your TEX run. You can move through the `dvi` file, and perhaps magnify sections of the output.

Printing. When you are ready to print, simply press the `Print` button in the `dotex` menu. This will run the `dvi` processor defined in the `xtmenu` script on the file just formatted and previewed. Any messages generated by the `dvi` driver appear in the shell window.

Once the windows are arranged, you are free to move between them, interacting with each as appropriate.

Arranging the windows. The figure in the appendix shows a typical arrangement of the windows. The previewer window is the largest with the other windows atop it. The shell window is kept small and unobtrusive near the bottom of the screen. The `dotex` menu is also near the `Control` menu of the editor, so that writing and TEXing require minimal mouse movement. Note that to preserve screen space it is sometimes useful to iconify both the shell and previewer window. In this arrangement, the focus is on the edit window and the `dotex` menu. The usual interaction is to make changes in the edit window, press the `TeX` button, and then view the file by pressing the mouse on the iconfied previewer window. Alternatively, the file can be previewed by bringing the large previewer window to the front of window stack. When the view is no longer needed, the previewer window can be pushed to the bottom of the stack.

The use of the X Window *window manager*, (Gettys, pages S2/49 – S2/52) is important to the use of `Dotex`, since the window manager's job of window sizing, movement and arrangement effects the productive use of the system. At a minimum the window manager is used to *raise* (place a window on the top of the stack) and *lower* (push a window to the bottom of the stack) the previewer window. All of these actions force `xdvi` to re-read the `dvi` file and present a fresh display.

Any standard X Window window manager will work with `dotex`, but since the predominate actions are raising, lowering and iconifying windows, a minimalist setup with the `twm` window manager (Querica and O'Reilly, chapters 3 and 10), works well. This setup is based the work of Pike (page 284, 1988) and presents a simple pop-up menu of window management functions that look like this:

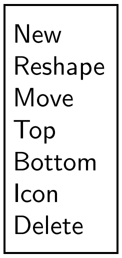


Figure 4: Window Management Popup

* Some implementations do not correctly handle errors during a backgrounded run.

These commands in the `twm` startup file define the setup:

```
NoTitle
Button3 = :all: f.menu "winmgr"
menu "winmgr"
{
    "New"        !"xterm  -ls &"
    "Reshape"    f.resize
    "Move"       f.move
    "Top"        f.raise
    "Bottom"     f.lower
    "Icon"       f.iconify
    "Delete"     f.delete
}
```

Figure 5: `twm` Window Management Definitions

Benefits

Prototyping typographic effects. The arrangement just presented has many benefits over the traditional TeX interaction model. For example, it facilitates the rapid prototyping of different typographic effects before final printing. As a simple example, to see the page-breaking effects of altering a parameter such as page width, the steps are:

- go to the editor window, and the add the change, for example, `\hsize=3in`,
- write the edit buffer,
- go to the `dotex` menu, press the `TeX` button, and
- then expose the previewer window.

If the editor supports the undo function, the change can be easily backed off if the desired effect did not meet expectations.

Network usage. `Dotex` runs in the X Window system so it is inherently network-based. This means that the appropriate hardware can be used for a particular document preparation task. For example, it is possible to run TeX on fast machine and at the same time use a different machine that has fast graphics for previewing.

Reduced loading. Once all of the tools are loaded and on-screen, they remain available to be used when needed, avoiding repeated startup and shutdown.

Rapid correction of errors. Because the error context from the log window and the editor are both visible, it makes error correction easier, not to mention having decreased overhead over the traditional method of having TeX spawn a new instance of the editor. The normal mode of operation becomes the more productive edit-TeX-preview instead of edit-TeX-print.

Enhancements

The window shell `wish` (Ousterhout, 1991) could replace `xtmenu` as the front-end to TeX and associated tools. `Wish` provides a richer widget set as well as the ability to define behavior of applications with the `Tcl` language. For example, instead of relying on shell scripts to provide the edit-format integration, a `Tcl`-aware editor such as `mx` could communicate directly via the `send` (Ousterhout, 1990) command with a `Tcl`-aware previewer.

Related Work

Pike (1984) describes an edit-format-preview tool built on the Blit window system, `troff` and UNIX pipes. This scheme has the advantage of immediate feedback — as soon as the file is written, it is immediately formatted and presented. Pike's approach is to use the operating and window system facilities to build the tool without changing the basic tools.

VorTeX (Chen 1988, pages 133 – 152) takes the approach of an incremental formatter/editor system using TeX and Emacs. The system provides two distinct but integrated views into the document, allowing direct manipulation of both.

Availability

All of the tools needed to implement `dotex` are publically available on the Internet.

Both `xdvi` and `xtmenu` have been posted to the USENET newsgroup comp.sources.x, and are archived on ftp.uu.net in

- `packages/X/contrib/xdvi.tar.Z`, and
- `packages/X/contrib/xtmenu\_1.1.tar.Z`.

`xdvi` is also available on export.lcs.mit.edu in the file `contrib/xdvi.tar.Z`.

The mouse based editor, `mx` is available on sprite.berkeley.edu in the file `tcl/mx.tar.Z`

Bibliography

Chen, Pehong, "A Multiple-Representation Paradigm for Document Development", *Technical Report UCB/CSD 88/436*, University of California, Berkeley, July 1988.

Gettys, Jim, Philip L. Karlton, and Scott McGregor, "The X Window System, Version 11", *Software Practice and Experience*, vol. 20, no. S2, pages S2/35 – S2/67, October, 1990.

Kernighan, Brian W. and Rob Pike, *The UNIX Programming Environment*, Englewood Cliffs, NJ: Prentice Hall, 1984.

Ousterhout, John K., "Tcl: An Embeddable Command Language", *Proceedings of the Winter 1990 USENIX Conference*, Washington, D.C., January 22-26, 1990.

Ousterhout, John K., "An X11 Toolkit Based on the Tcl Language", pages 105-115. *Proceedings of the Winter 1991 USENIX Conference*, Anaheim, CA,

Pike, Rob, "The Blit: A Multiplexed Graphics Terminal", *Bell Labs Tech. J.*, vol. 63, no. 8, Part 2, pages 1607 – 1631, 1984.

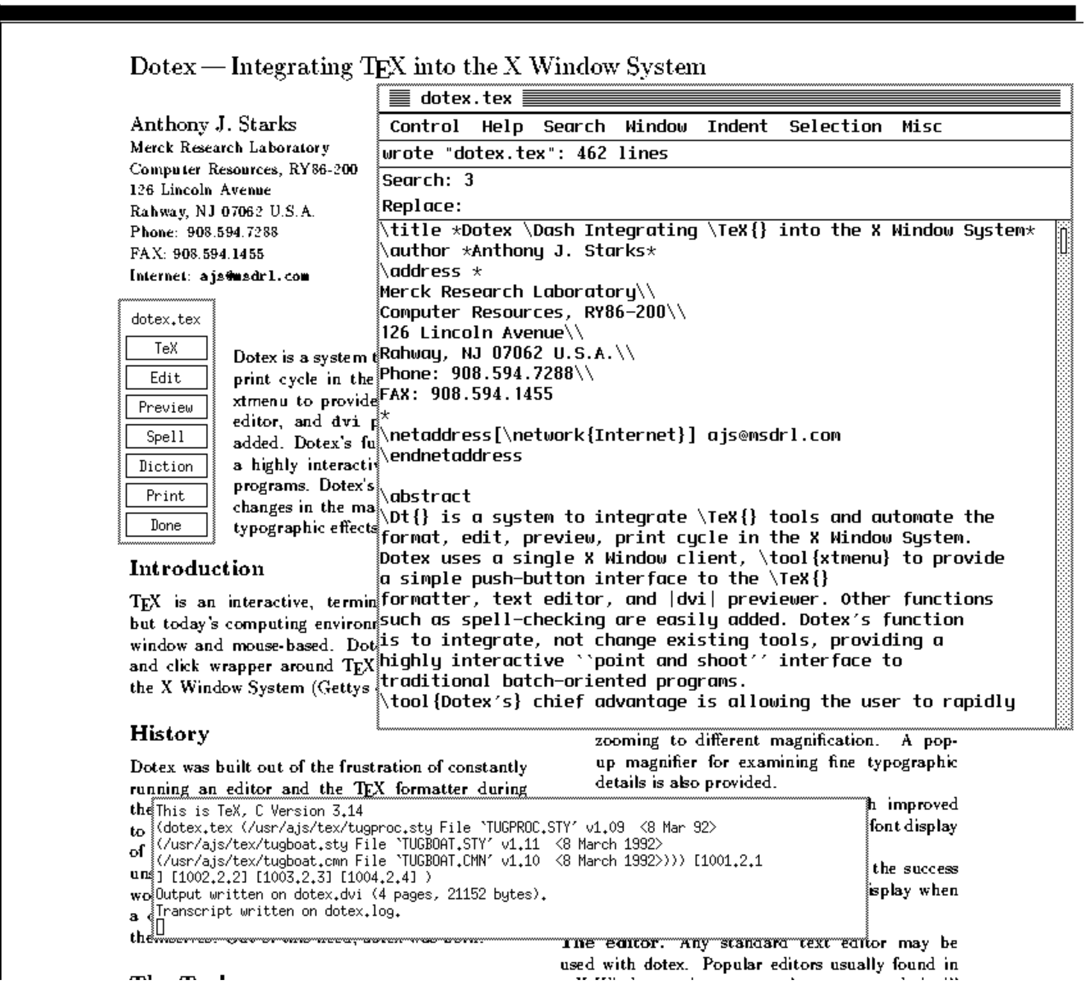
Pike, Rob, "Window Systems Should Be Transparent", *USENIX Computing Systems*, vol. 1, no. 3, pages 279-296, Summer, 1988.

Quercia, Valerie and Tim O'Reilly, *X Window System User's Guide, for X11 R3 and R4 of the X Window System*, Sebastapol, CA: O'Reilly and Associates, Third Edition, 1990.

Taylor, William, "xtmenu — An X Windows Menu Program", Version 1.0, *comp.sources.x*, vol. 13, xtmenu, June 17, 1991.

Vojta, Paul, et al, "xdvi — DVI Previewer for the X Window System", *comp.sources.x*, vol. 17, xdvi, March, 1992.

Appendix I — A Sample Session



Dotex — Integrating TEX into the X Window System

Anthony J. Starks
Merck Research Laboratory
Computer Resources, RY86-200
126 Lincoln Avenue
Rahway, NJ 07062 U.S.A.
Phone: 908.594.7288
FAX: 908.594.1455
Internet: ajs@msdrl.com

Dotex is a system t
print cycle in the
xtmenu to provide
editor, and dvi p
added. Dotex's fu
a highly interacti
programs. Dotex's
changes in the ma
typographic effects

Introduction

TEX is an interactive, termin
but today's computing environ
window and mouse-based. Dot
and click wrapper around TEX
the X Window System (Gettys

History

Dotex was built out of the frustration of constantly
running an editor and the TEX formatter during
the
to
of
uns
wo
a
the

zooming to different magnification. A pop-
up magnifier for examining fine typographic
details is also provided.

h improved
font display

the success
isplay when

The editor. Any standard text editor may be
used with dotex. Popular editors usually found in

```
wrote "dotex.tex": 462 lines
Search: 3
Replace:
```

```
\title *Dotex \Dash Integrating \TeX{} into the X Window System*
\author *Anthony J. Starks*
\address *
Merck Research Laboratory\\
Computer Resources, RY86-200\\
126 Lincoln Avenue\\
Rahway, NJ 07062 U.S.A.\\
Phone: 908.594.7288\\
FAX: 908.594.1455
*
\netaddress[\network{Internet}] ajs@msdrl.com
\endnetaddress

\abstract
\Dt{} is a system to integrate \TeX{} tools and automate the
format, edit, preview, print cycle in the X Window System.
Dotex uses a single X Window client, \tool{xtmenu} to provide
a simple push-button interface to the \TeX{}
formatter, text editor, and |dvi| previewer. Other functions
such as spell-checking are easily added. Dotex's function
is to integrate, not change existing tools, providing a
highly interactive ``point and shoot'' interface to
traditional batch-oriented programs.
\tool{Dotex's} chief advantage is allowing the user to rapidly
```

```
This is TeX, C Version 3.14
(dotex.tex (/usr/ajs/tex/tugproc.sty File `TUGPROC.STY' v1.09  <8 Mar 92>
(/usr/ajs/tex/tugboat.sty File `TUGBOAT.STY' v1.11  <8 March 1992>
(/usr/ajs/tex/tugboat.cmn File `TUGBOAT.CMN' v1.10  <8 March 1992>))) [1001.2.1
] [1002.2.2] [1003.2.3] [1004.2.4] )
Output written on dotex.dvi (4 pages, 21152 bytes).
Transcript written on dotex.log.
```

Figure 6: A `dotex` session

This typical arrangement shows the shell window at the bottom, with the edit window atop the large previewer window. The `dotex` menu is to the left of the edit window.

Appendix II — Customizing dotex scripts

This script is more elaborate than the simple one presented above, and adds spell and grammar checking. Note the use of horizontal orientation which is useful when there are many menu items. Also note the blank label used here to provide visual separation of the functions.

```
file=$1
xtmenu -horizontal -noquit -stdin <<!
"$file.tex" !label
TeX         %tex $file.tex
Edit        %mx $file.tex
Preview     %xdvi $file.dvi&
""          !label
Spell       %xterm -e ispell $file.tex
Diction     %striptex $file.tex | diction
Print       %dvips $file
Done        !exit
!
```

Of course `dotex` is not limited to plain TEX; this example runs LATEX and adds special BIBTEX support. Also added is an additional button to view the log file.

```
file=$1
xtmenu -noquit -stdin <<!
"$file.tex" !label
LaTeX       %latex $file.tex
BibTeX      %bibtex $file
Clean       %rm -f $file.aux $file.bbl && echo Work files cleaned.
Edit        %mx $file.tex
Log         %mx $file.log
Preview     %xdvi $file.dvi&
Print       %dvips $file
Done        !exit
!
```

This script is used to automate the creation of documents with included figures. Buttons are defined to popup the figure drawing tool `xfig` and to translate the figure file to LATEX.

```
file=$1
xtmenu -horizontal -noquit -stdin <<!
"$file.tex" !label
LaTeX       %latex $file.tex
Edit        %mx $file.tex
Log         %xterm -e less $file.log
Figure      %xfig $file.fig&
fig->TeX    %fig2dev -Llatex $file.fig >${file}-fig.tex
Preview     %xdvi $file.dvi&
Print       %dvips $file
Done        !exit
!
```

Another way to run `xtmenu` is with multiple *menu files*, each tailored to a different formatting situation. In this case, the `dotex` shell script might look like this:

```
#
#  dotex -- automate TeX and tools
#  (multiple menu file version)
#
#  Usage:  dotex [-l] [-p] [-b] file
#
mdir=/usr/local/lib/xtmenus
option=$1
TeXfile=$2
export TeXfile
case $option in
-l)     xtmenu -noquit -m $mdir/latex.xtm&;;
-p)     xtmenu -noquit -m $mdir/pictex.xtm&;;
-b)     xtmenu -noquit -m $mdir/bibtex.xtm&;;
*)      xtmenu -noquit -m $mdir/plain.xtm&;;
esac
```

Figure 7: A `dotex` script with multiple menu files

where the different menu files are stored in a special directory `/usr/local/lib/xtmenus`. All of the menu files use a common variable, `$TeXfile` when referring to the TeX source file. The different styles of TeX use are invoked by using a different option character, as in:

```
dotex -l file
```

to run the LaTeX specific setup.

Appendix III — The newer program

```
/*
 * newer -- compare the age of two files
 *
 *
 * Usage: newer file1 file2
 *
 * A zero status is returned
 * if and only if file1 was modified after file2.
 *
 *
 *
 */

#include <sys/types.h>
#include <sys/stat.h>

main(argc, argv)
        char          **argv;
        int             argc;
{
        struct stat     s1, s2;

        if (argc != 3)
                exit(1);

        if (stat(argv[1], &s1) < 0)
                exit(1);

        if (stat(argv[2], &s2) < 0)
                exit(1);

        if (s1.st_mtime >= s2.st_mtime)
                exit(0);
        else
                exit(1);
}
```