EZTex: A Graphical User Interface System for \LaTeX

Ravi K. Kosaraju
Western Michigan University

Follow this and additional works at: https://scholarworks.wmich.edu/masters_theses

Part of the Computer Sciences Commons

Recommended Citation
https://scholarworks.wmich.edu/masters_theses/865
EZ\TeX{}: A GRAPHICAL USER INTERFACE SYSTEM FOR \LaTeX{}

by

Ravi K. Kosaraju

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
December 1992
EZT\TeX: A GRAPHICAL USER INTERFACE SYSTEM FOR \LaTeX

Ravi K. Kosaraju, M.S.

Western Michigan University, 1992

In this thesis, we have developed a Graphical User Interface (named EZT\TeX) for the popular typesetting program \LaTeX. Our work has been motivated by the lack of an integrated system on the SUN workstations that lets simultaneous viewing and editing of \LaTeX documents.

In EZT\TeX, the \LaTeX macros are represented in the form of pop-up menus and icons which have “almost” eliminated the need for referring to the \LaTeX manuals. Making use of the logical design aspect of \LaTeX, EZT\TeX provides a unique blend of visual and logical design principles. EZT\TeX provides options for using various programs such as \textit{xdvi}, \textit{dvipage}, \textit{spell} which are useful in designing a \LaTeX document.
ACKNOWLEDGMENTS

Although it is not easy to attribute my success to any one person during my study here at Western Michigan University, I would like to list a few who contributed in the completion of this thesis.

First, I thank my advisor, Dr. Naveed A. Sherwani. His guidance and patience throughout my study enabled me to pursue my goals with utmost confidence and enthusiasm.

Secondly, I extend my sincere appreciation to the members of my thesis committee, Dr. Fred Boals and Dr. Ajay Gupta, for their supervision and support. I am also thankful to Dr. Kenneth Williams for all the support he provided me as a Graduate advisor during my study years.

Last but not the least, I would like to thank all the faculty and fellow students in the Department of Computer Science at Western Michigan University, Kalamazoo who have ever provided opportunity, encouragement, and support to me.

Ravi K. Kosaraju
To my father,
Late Dr. K Reddiah.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
EZTex: A graphical user interface system for \LaTeX

Kosaraju, Ravi Kumar, M.S.
Western Michigan University, 1992
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................................. ii

LIST OF FIGURES ........................................................................ vii

I. INTRODUCTION ...................................................................... 1

1.1 Main Features of \LaTeX{} ......................................................... 3

1.2 Review of Related Literature .................................................. 4

1.2.1 Instant\TeX{} ...................................................................... 4

1.2.2 Salient Features of Instant\TeX{} ......................................... 4

1.2.3 \TeXit{} ............................................................................. 5

1.2.4 Salient Features of \TeXit{} .................................................. 6

1.2.5 \TeXmenu{} ....................................................................... 6

1.2.6 Salient Features of \TeXmenu{} ............................................ 7

1.3 Objectives of Thesis Project .................................................. 8

1.4 Thesis Organization ............................................................ 9

II. SALIENT FEATURES OF EZ\TeX{} .......................................... 10

2.1 The EZ\TeX{} Graphical User Interface .................................... 11

2.2 Text Entry and Editing Features .......................................... 11

2.2.1 Text Entry ...................................................................... 11

2.2.2 Pattern Searching .......................................................... 12

2.3 Document Properties .......................................................... 12

2.3.1 Mathematical Functions and Symbols ............................... 13

2.3.2 Display Current Document Structure ............................... 13

2.3.3 Running \LaTeX{} and Previewing \texttt{dvi} File .................. 15

2.3.4 Including External Files ................................................ 15

iii
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4 Document Layout</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2.4.1 Local and Global Error Detection</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2.4.2 Numbering</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>2.4.3 Matched Pairing of Commands</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>2.4.4 Creating Bibliography</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2.5 Drawing in EZT_{p}X</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>III. PROGRAM DESIGN AND METHODOLOGY</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>3.1 An Introduction to the X Window System</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>3.1.1 What is X?</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>3.1.2 The X Server</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3.1.3 Windows and Events</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3.1.4 Xlib - The Assembly Language of X</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3.1.5 X Toolkits - The High-Level Languages of X</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3.2 The OPEN LOOK\textsuperscript{TM} Graphical User Interface</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3.2.1 What is a Graphical User Interface?</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3.3 Implementation of the EZT_{p}X Interface</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3.3.1 An Introduction to OPEN LOOK\textsuperscript{TM} Toolkits</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>3.3.2 The Class Hierarchy of OPEN LOOK Widgets</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>3.3.3 An Overview of Windows and Their Functions</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>3.3.4 Software Development and Programming Techniques</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>IV. A USER'S MANUAL FOR EZT_{p}X</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>4.1 Getting Started</td>
<td>35</td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents – Continued

CHAPTER

4.2 The File Menu ................................................................. 36
  4.2.1 Creating a New Document ............................................. 36
  4.2.2 Loading a Document From the Disk ............................... 36
  4.2.3 To Clear the Current Text Buffer ................................. 38
  4.2.4 Saving the Contents of Text Window ......................... 38
  4.2.5 Saving the Text Buffer Onto a New File ....................... 38
  4.2.6 Quitting EZTeX ......................................................... 39

4.3 The Edit Menu ............................................................... 39

4.4 The Properties Menu .................................................... 39
  4.4.1 The Math Menu .......................................................... 40
  4.4.2 The Symbols Menu ...................................................... 40
  4.4.3 Automatic File Backup ................................................ 41
  4.4.4 Displaying the Current Document Structure ................ 46
  4.4.5 Processing the “Current Document” .............................. 47

4.5 Document Properties .................................................... 48
  4.5.1 Document Preamble ..................................................... 49
  4.5.2 Selecting Text “Size” and “Style” for the Document .......... 49
  4.5.3 Including External Files ............................................. 49

4.6 The Layout Menu .......................................................... 50
  4.6.1 Sectioning ................................................................. 50
  4.6.2 Sentences ................................................................. 51
  4.6.3 Environments ............................................................ 51
  4.6.4 Declarations ............................................................. 52
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6.5 Defining, Cross-Referencing and Indexing</td>
<td>52</td>
</tr>
<tr>
<td>4.6.6 Reading Input and Writing Output Interactively</td>
<td>52</td>
</tr>
<tr>
<td>4.6.7 Creating Bibliography</td>
<td>52</td>
</tr>
<tr>
<td>4.7 Drawing in EZTP(X)</td>
<td>53</td>
</tr>
<tr>
<td>4.8 The Format Menu</td>
<td>54</td>
</tr>
<tr>
<td>4.8.1 Page Format</td>
<td>54</td>
</tr>
<tr>
<td>4.8.2 Paragraph Format</td>
<td>54</td>
</tr>
<tr>
<td>4.8.3 Numbering</td>
<td>54</td>
</tr>
<tr>
<td>V. CONCLUSIONS AND DIRECTION OF FUTURE RESEARCH</td>
<td>64</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>66</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1. Text Manipulation Functions ............................................................... 14
2. Document Properties .............................................................................. 14
3. The Document Format of a Document Using the \texttt{xdvi} Program. 16
4. Output Produced by Running \LaTeX on the Current Document. 17
5. The \textit{"setcounter"} and \textit{"addtocounter"} Environments .......... 20
6. An Example Bibliography. ................................................................. 21
7. The Client-Server Model in X .............................................................. 25
8. Diagram Illustrating the Use of \textit{"New"} Menu Option. ............... 38
9. The \textit{"Load"} Menu Option and the Associated Pop-up Window. 39
10. The Various Features Offered by the \textit{"Edit"} Menu Option .......... 41
11. Various Command Options Available Under \textit{"Math"} Menu Option. 42
12. Categories of Mathematical Symbols Available in EZTeX. .......... 44
13. List of \textit{"Binary Operation Symbols"} Available in EZ\TeX. 45
14. List of \textit{"Arrow Symbols"} Available in EZ\TeX. ....................... 45
15. List of \textit{"Miscellaneous Symbols"} Menu ........................................ 46
16. List of \textit{"Relation Symbols"} Available in EZ\TeX. ..................... 46
17. List of \textit{"Variable-Sized Symbols"} Available in EZ\TeX. .......... 47
18. List of \textit{"Log-like-functions"} Available in EZ\TeX. .................... 47
19. List of \textit{"Greek Symbols"} Available in EZ\TeX. .......................... 48
20. The Automatic File Backup Utility ...................................................... 49
21. The Four Options Available in \textit{"Document"} Menu ...................... 51
22. The Three Options Provided by the \textit{"Splitting Input"} Menu Option. 53

vii

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
List Of Figures – Continued

23. The Thirteen Options Available Under the “Layout Menu.” . . . 59
24. Options Available Under the “Sectioning” Menu Option. . . . 60
25. The “Sentences” Menu Option and Its Associated Submenu. . . 61
26. The Two Different Modes of Spacing Allowed in EZE\TeX. . . . . 61
27. The Various “Special” Characters Available in Sentences Menu. . 62
28. The Various Environments Supported by EZE\TeX. . . . . . . . . . 63
29. The “Declarations” Menu and Its Options. . . . . . . . . . . . . . 64
30. The “Define” and “Cross References” Menus and Their Options. . 64
31. The “Draw” Option in EZE\TeX With an Example Listing of xfig Files. 65
32. The “Format” Menu and the Submenus Associated With It. . . . 66
33. A List of Options Available Under the “Boxes” Menu Option. . . 66
CHAPTER I

INTRODUCTION

With the advent of sophisticated computers and advanced techniques in information processing, computers have become an essential part of our lives. Almost every known profession that humans are involved in, requires the use of computers in one way or the other. Not only are computers used for data processing applications, a variety of industries such as desktop publishing, typesetting, multimedia etc. are heavily dependent on them.

During the olden days when computers were not as commonly used as they are now, typesetting was performed using the traditional approach. Thus if an author wanted to publish a manuscript, he would use a typewriter to type the text and provides it to the publisher. Once the typewritten script is approved, a person responsible for formatting this manuscript designs the layout of the manuscript. This involved providing specifications such as the length of the document, the style of type to use, the top and bottom margins etc. These specifications are given in written form to the typesetter who is responsible for typesetting the final manuscript. Typesetting during those days was a tedious process. For each page in the manuscript, a metal sheet is created in which the text is imprinted. The process of creating a single document used to take days, sometimes even weeks.

Nowadays, the same procedure of typesetting is accomplished using computers. The user types in the document on a computer and typically uses a word-processor to create the final output of the document. Though there are a number of wordprocessing packages available in the market, most of them have a major drawback, they are not typesetters. These documents do provide a good visual
interface for the user for typing the documents but they have limited features and flexibility. This type of design is known as a visual design. The biggest advantage of using such a design is that the user is able to see the output while he/she is typing. The disadvantage in using such a design is that the user believes that if the document appears to look good on the computer screen, it is well designed. This may lead to poorly designed documents.

On the other hand there are a few software packages available that use what is known as a logical design. The advantages of using a logical design approach are as follows.

1. It enhances the use of sound typographic methods.
2. Well designed documents are created.
3. The author has more flexibility in formatting the document.

Modern day typesetting is not only used for printing books and manuscripts but also used in creating a project report for a class, writing journal papers, designing brochures etc. Currently there is a typesetting program available in the market called $\TeX$ designed by Donald Knuth. $\TeX$ is a sophisticated program that produces high-quality typesetting, especially for designing mathematical text. Internally $\TeX$ is designed as a collection of macros that are used for formatting and typesetting a document. The advantages of using $\TeX$ as a typesetting document are several.

1. Among all the programs that are used for typesetting or wordprocessing, $\TeX$ is one of the few programs that works on all existing computers.
2. $\TeX$ files could be easily sent to different people on different systems. These files could be run on these machines without any changes to the file.
3. $\TeX$ creates a device independent file known as a dvi file that could be processed on any computer without modification.
4. Most of the journals and conferences consider $\TeX$ as their standard.
With all the flexibility for typesetting, $\TeX$ does have a major disadvantage. Designing a document using $\TeX$ is not very easy, especially from a layman's point of view. Firstly, the user needs to remember the various macros for designing a document. Secondly, the formatting commands are quite confusing for the end user.

To curb these disadvantages the $\LaTeX$ document preparation system was designed. $\LaTeX$ adds a collection of commands to $\TeX$ that simplify typesetting. In other words, $\LaTeX$ acts as a shell on top of $\TeX$ that lets the user concentrate more on the structure of the text rather than the commands involved in typesetting the document.

1.1 Main Features of $\LaTeX$

1. One of the fundamental features in $\LaTeX$ is the idea of a document style that determines how a document needs to be formatted. e.g., it is the responsibility of $\LaTeX$ to generate the equation numbers of the equations with the document style specifying the type of numbering to use.

2. The user needs to concentrate more on the writing part of the document rather than be concerned with the formatting commands.

3. The $\LaTeX$ commands that are typed in are translated to low level $\TeX$ commands.

4. New capabilities to $\LaTeX$ could be added by using the lower-level $\TeX$ commands.

5. A device-independent file or dvi file is created by $\TeX$ that is portable to any computer system without any modifications.

6. $\LaTeX$ provides the user with flexibility in the typesetting process. Since the document design is performed in a logical fashion, it is easy to modify the document once it is typed.
1.2 Review of Related Literature

There are a number of programs available in the market that offer user interfaces for \TeX{} and \LaTeX{}. Most of these are available as public domain software which could be obtained free of cost. These programs could be obtained from various \texttt{ftp} sites such as \texttt{export.lcs.mit.edu}, \texttt{sonata.cc.purdue.edu} etc. One drawback in using these programs is that they do not work on all computer systems. Following is a brief description of these programs and the computer systems they work on.

1.2.1 Instant\TeX{}

This program is written for the NeXT computer. This is a user friendly interface to \TeX{} for the NeXT computers. It is a completely menu-driven program that uses various pull-down menus and buttons to accomplish typesetting. Unfortunately, it is available only on the NeXT and hence could not be used on other computer systems. Instant\TeX{} is available at \$150 for commercial, government and private users. The price for academic usage is \$120 (Students - \$45.)

1.2.2 Salient Features of Instant\TeX{}

1. Various options are provided for formatting the text file. Apart from \TeX{} and \LaTeX{} formats, there are a few other formats defined in the program such as Phyzzx, Harvmac, Ams\TeX{} etc. The formatting can be accomplished by just a few keystrokes.

2. The document format of the \TeX{} files can be determined automatically.

3. One of the interesting features of this program is that if \TeX{} notices an error in the file begin processed, just by pressing a key, Instant\TeX{} will go to the line that contains the error. This makes error detection and correction a much simpler task.
4. An important feature of *InstantTeX* is that it has an "Instant" mode, that allows one to process the file (TeX) automatically when it is being written and preview the results in TeXview almost simultaneously. The output appears a few keystrokes later. This makes it easy to write equations, macros and for adjusting the document format.

5. Instant mode can also be used with multiple files. It is also possible to process only a portion of a text instead of the whole document.

6. Among other features, *InstantTeX* provides an option called TeXPalette, which contains buttons corresponding to Greek letters, Mathematical operators, Relations and other commonly used TeX symbols. By selecting any of these buttons, the program writes the corresponding TeX or LaTeX code to the text file. Many different LaTeX environments are also provided in the form of buttons. There are buttons defined for creating complex mathematical formulas and equations, commonly used arrays and determinants etc.

7. Pictures could be inserted into a document by just dragging them into the TeX file window. This feature is one of the strong points of *InstantTeX*. Also pictures from other applications such as the Adobe Illustrator, Mathematica and Draw could be very easily imported into the TeX file.

8. A spell checker is provided that can spell-check a LaTeX document. The spelling-checker provided on the NeXT computer is used to accomplish this.

1.2.3 TeXit

TeXit was designed by Paul A. Griffin at the University of Florida, Gainesville, FL. It was compiled under NEXTSTEP 2.1 and is available on the NeXT computer.

TeXit was designed to reduce the time cycling between the editor and the previewer when a TeX document is being written. There are three services that TeXit provides. These are as follows.
1. **$\TeX$ File** – Typesets the entire document (or file.)

2. **$\TeX$ Pasteboard** – Used for typesetting the ASCII pasteboard, including a user defined header and end-command.

3. **$\TeX$ Selection** – Typesets selected (highlighted) text in a document, including a user defined header and an end command.

### 1.2.4 Salient Features of TeXit

1. Commands are executed by either pressing one of the icon-buttons from TeXit’s main window or by selecting a command from the “Commands” menu.

2. By choosing the option “$\TeX$ file”, the typesetting of the file is invoked as a subprocess of the current process. If the file happens to be within the user’s home directory, the output file is generated in the same directory, else it is generated in “\tmp” directory.

3. The “$\TeX$ Pasteboard” option is useful in running $\TeX$ on whatever is in the ASCII pasteboard.

4. If the “Previewer” command preference is set, TeXit automatically sends TeXview the typeset document and opens TeXview for viewing (TeXview is the dvi file previewer for NeXT.)

5. TeXit implements $\TeX$’s “e” command, i.e., typing an “e” in the window where $\TeX$ is running will edit the file at the offending line. This function is performed through the use of the “Edit” application on NeXT.

6. Using the “Goto Line” command, one could easily edit files with non-prompting errors. For example, “\overfull hbox” etc.

### 1.2.5 TeXmenu

TeXmenu is a program written by Harald Schlangmann and is available on the NeXT computer.
1.2.6 Salient Features of TeXmenu

1. Provides command options for using plainTeX, \LaTeX, \SLiTeX or Am-
sTeX.

2. Lets the user view the results using TeXview (similar to the TeXit
program.) The auxiliary files generated could also be viewed using one of the
command options.

3. Editing of \TeX files is performed with the NeXT editor “Edit”.

4. Provides an option to create bibliographic entries using the BIB\TeX
environment.

5. Lets the user customize the \TeX environment.

6. TeXmenu allows the user to work on multiple files simultaneously, par-
ticularly with file structures generated by the \texttt{\input} statements of
\LaTeX and \TeX.

7. All settings and files used in a session are saved into a defaults database
so that the next time the user evokes TeXmenu, these settings are applied.

8. There is a parsing option available in TeXmenu that is used for parsing
the \TeX files.

There are certain drawbacks in using the TeXmenu program. Following
are a few bugs that may occur while using TeXmenu.

1. Certain control sequences such as Control-D do not work when TeXmenu
is running. This problem could be solved by typing “null” if \TeX asks for a file
name.

2. The parser does not work very well with \TeX. Also, if the user needs
to parse a \LaTeX file, he/she has to insert certain text that is unique to the \LaTeX
format. For example, he needs to insert the command “documentstyle” in the
preamble for a \LaTeX document. Since this command is unique to \LaTeX, the
parser is able to distinguish it from the other documents.
1.3 Objectives of Thesis Project

The main objective of this project is to develop a *Graphical User Interface* for processing \LaTeX{} files. The popularity of \LaTeX{} as a typesetting standard not only has increased manyfold in the professional field but it is also immensely popular among the academic community and has become a de-facto standard for publishing manuscripts and papers. One major obstacle in using \LaTeX{} or \TeX{} is that document creation process could be very cumbersome. Since \LaTeX{} is nothing but a collection of commands used for typesetting of documents, a number of such commands are available for formatting the documents. Remembering all these commands is very difficult. On top of this the \LaTeX{} manual doesn’t make matters easy either. It is a struggling experience for an end user to get a complete grasp of the commands used in \LaTeX{}. As a result, most of the users spent a substantial amount of time in learning \LaTeX{} and the various command structures in it.

To overcome these drawbacks, we have attempted to design a user-friendly system that provides a *Graphical User Interface* for \LaTeX{}. In other words, the user need not be concerned about the command structure of most of the \LaTeX{} commands and is presented with a Macintosh-like user interface in which he/she needs minimum effort in creating a \LaTeX{} document. The program is designed using the C programming language and is built on an X-Windows/OPENLOOK platform. This interface is provided in the form of pull-down and pop-up menus and buttons that the user needs to select to create a document from scratch.

In this project we have tried to create an atmosphere which is a combination of both the *logical* and the *visual* design aspects of document design. This program is one of the few of its kind in the market and we hope that it will become a handy tool for people using \TeX{} and \LaTeX{} for designing their manuscripts.
1.4 Thesis Organization

This thesis is organized into five chapters. In Chapter II, a brief introduction to the X-Windows and Openwindows system is presented. This is the platform on which this thesis project is based on. Chapter III provides a detailed description about the functionality of the various command options available in EZTeX. Chapter IV describes the internal organization of the program and provides detailed description about the implementation of the program. We conclude this report by discussing the possibility for future extensions to this program.
CHAPTER II

SALIENT FEATURES OF EZ\TeX

As mentioned in the previous chapter, the purpose of this project was to design a Graphical User Interface for the popular typesetting program \LaTeX. This interface is built on an OpenWindows/OPEN LOOK\textsuperscript{TM} graphical interface platform. We used the C programming language to write this interface. A major portion of this program was written using \textit{X Toolkit programming techniques} (To be explained in chapter IV.) Since this interface is based on the X Toolkit library system and is not based on the \textit{Xlib library}, portability of this program to other computer systems is not very easy. This is due to the fact that the X Toolkits utilities are dependent on the graphical interface that exists on a computer. Hence a program that is written using the X Toolkits on a SUN SPARCstation would contain OPEN LOOK\textsuperscript{TM} system calls. If this program has to be ported to a different system that has a graphical interface other than the OpenWindows/OPEN LOOK\textsuperscript{TM} interface (e.g., OSF/Motif), these calls have to be modified. We have tried to provide a Macintosh-like graphical user interface for our program. A majority of \LaTeX\ commands are included in this package in the form of menu buttons and popup windows. We chose to exclude those commands that are not commonly used. These commands may be added in the future extensions to this program. This is a simple and easy to use interface and could be useful to people who are not familiar with \TeX or \LaTeX.

In the following sections, we describe the various features offered by EZ\TeX. We also compare and contrast our program to the other existing software programs that offer similar capabilities.
2.1 The \texttt{EZ\TeX} Graphical User Interface

As mentioned in the previous section, the \texttt{EZ\TeX} program interface is a combination of several menus and menu options. To execute a command a user has to select a particular menu button. In total there are more than 150 menu options for the user to choose from. These options could be divided into the following categories. Each menu option in \texttt{EZ\TeX} belongs to one of these three categories.

1. \textit{Text Manipulation} – Used for text input, text editing, searching text patterns and other text manipulation features.

2. \textit{Entering \LaTeX\ commands} – These menu options when selected, insert the corresponding \LaTeX\ code into the text window.

3. \textit{Executing other programs} – Used for executing programs (such as \texttt{xdvii}, \texttt{pageview}) that are useful in creating \TeX\ documents.

2.2 Text Entry and Editing Features

The "File" and "Edit" menu options on the main menu provide various command options for manipulating text. Figure 1 illustrates some of these options.

2.2.1 Text Entry

1. \textit{Creating new documents} – \texttt{EZ\TeX} provides a text window (attached to the Main window) for entering text. A user may start a new document or load a \LaTeX\ file into the Text Window.

2. \textit{Loading a disk file} – The Load menu option lets the user load a file from the current directory or any directory that the user specifies.

3. \textit{Document preamble} – When a new document is selected by choosing the "New" menu option, A popup menu consisting of a list of commonly used
declarations for the document *preamble* is brought up. This preamble is automatically inserted at the beginning of the document. The *Document Preamble* submenu option under the *Document* menu option lets the user set the *preamble* declaration statements during the text entry phase.

4. *Saving text* - The "Save" and "Save As" menu options are provided for saving the current text buffer onto a disk file. The program performs error checking before saving the file and informs the user in case of an error while saving.

2.2.2 Pattern Searching

Under the *Edit* menu option, command options for text pattern searching are provided.

1. *Searching for text patterns* - Using text buffers, we have implemented the text searching and replace options that are provided by most of the text editors such as (*vi, textedit etc.*) The "Find", "Goto Line" and "Select All" command options are used for searching and replacing text patterns.

2. *Spelling checker* - The "spell" program available in UNIX can be executed by selecting the "Spell" command option. The user enters the input and output file names for running *spell*. After executing *spell*, the user could view the output generated by *spell* in a different window. This is achieved by selecting the "Open spell file" command option.

2.3 Document Properties

Under the "*Properties*" menu option, we have provided a number of options for setting certain document properties. Following is a list of these options along with their functionalities. Figure 2 presents some of the options that are provided by the "*Properties*" menu option.
2.3.1 Mathematical Functions and Symbols

Math functions – This option provides a list of commonly used math options (formula, fraction, arrays etc). By selecting any of these options, the corresponding \LaTeX commands are inserted into the text window at the current cursor location. If the command being inserted happens to be a declaration i.e., a command that does not contain any arguments, then the program places the cursor at the end of this command string. On the other hand if this command contains argument(s), then the cursor is placed inside the braces that encompass the argument. If the user inserts a command inside another command’s argument, the cursor is placed inside the innermost nesting of braces.

Symbols – A very useful option is the “Symbols” command option. This option is provided to generate the commonly used greek and mathematical symbols (delta, pi, plus-minus etc.) This feature eliminates the need to look up the \LaTeX commands for these symbols. We have provided the user with an option to exclude the $ symbols that are placed at the beginning and the end of a mathematical or greek symbol. This is helpful to the user if he/she wants to include several symbols in one line of text (which only requires one set of $ symbols.)

Automatic backup of text – While using this program, if due to some reason, the program or even the system crashes, the user should at least have a recent backup of the current text available. For this reason we have provided an automatic backup procedure that creates a backup of the current text contents in the text window. The contents of the text window are backed up periodically. An option is provided for the user to specify the time-period between two consecutive backups.

2.3.2 Display Current Document Structure

The user can view the document structure (The main headings in the document including sections, subsections etc) by selecting the option “Show Current
Figure 1. Text Manipulation Functions.

Figure 2. Document Properties.
Format. The structure could be viewed using different programs (\texttt{xdvi}, \texttt{dvipage}, \texttt{pageview}). Figure 3 shows the document structure of an example \TeX\ file using the \texttt{xdvi} program.

2.3.3 Running \LaTeX\ and Previewing dvi File

1. The "Process File" command option is used to process the current text in the text window using \LaTeX. The output of the processing is displayed in a separate window. Figure 4 illustrates an example of the output produced by running \LaTeX\ on the current text.

2. When \LaTeX\ detects an error in the text while processing the current text, the program detects this error and retrieves the line number where the supposed error has occurred. The cursor is then placed at the end of this line. This makes it easy for the user to edit the line where the error has occurred.

3. If the current text buffer (Text window) does not contain text, then the user receives a warning in the message window.

4. To preview the dvi file for the current text, the "Open dvi" command option is provided. We have used the \texttt{xdvi} program as the dvi file previewer. The "Reopen xdvifl" command option is used to "reopen" \texttt{xdvi} after running \LaTeX\ on the current document. The original \texttt{xdvi} program was modified such that the dvi file could be "reopened" by selecting the "Reopen xdvifl" button.

2.3.4 Including External Files

The "\texttt{\include}", "\texttt{\input}" and the "\texttt{\includeonly}" commands in \LaTeX\ could be inserted into the text window by selecting the "Splitting Input" command option. The user types in the file(s) to be included in a textfield window. The program then searches the specified directory to check if the listed file(s) is/are indeed available on the disk. An appropriate error message is produced if the files listed are missing from the directory.
1 Introduction
1.1 Types of data fragmentation

2 Data partitioning techniques

3 Vertical partitioning based on empirical objective functions and

3.1 Algorithms based on empirical objective functions
3.1.1 Affinity Among Attributes
3.1.2 Clustering of Attributes
3.1.3 Partitioning an object into Nonoverlapping Fragments
3.1.4 Partitioning an object into Overlapping Fragments
3.2.5 Partitioning into Primary and Secondary Fragments

3.2 Vertical Partitioning based on Cost Optimization

3.3 Graphical Technique for Vertical Partitioning
3.3.1 The Algorithm

Figure 3. The Document Format of a Document Using the xdvi Program.
A distributed database is a collection of different computers of a computing system, each of which also participates in the execution of the application. The decomposition of global relations is an important aspect of distributed database design. Emergency stop.

Figure 4. Output Produced by Running \LaTeX on the Current Document.
2.4 Document Layout

A number of command options are provided under the "Layout" menu option. Refer to Chapter IV for all the commands and declarations supported by E\LaTeX. Most of the commands and declarations that are part of this menu option are accomplished by simple text retrieval from a global database. Following is a list of some of the significant features of the "Layout" menu option.

2.4.1 Local and Global Error Detection

After the command selection and the text insertion of the corresponding \LaTeX command, an extensive error checking of the document is performed to detect obvious errors. The error checking process is performed in two phases:

1. Local Error Check – Local error checking is performed on the current line (the line that contains the cursor.) The current line is parsed from left to right and if a \LaTeX command is encountered it is checked for validity. All valid commands are stored in a global database. If the command is a valid \LaTeX command, the arguments for this command (if any) are checked for validity. If an error is found, appropriate error messages are displayed in the message window.

2. Global Error Check – A Global error check is made starting from the last line of the document going backwards to the beginning of document. The global check is to make sure that for every \texttt{\begin} declaration in an environment, there is a \texttt{\end} declaration. Also the order in which the environments are nested is checked. If an environment occurs inside another, their respective \texttt{\begin} and \texttt{\end} declarations should be properly nested.

2.4.2 Numbering

The two commands viz., \texttt{\setcounter} and \texttt{\addtocounter} could be written to the text window by respectively choosing the \texttt{\setcounter} and \texttt{\addtocounter} commands from the "Numbering" menu option. When chosen, each of
these buttons causes a pop-up window to appear that contain various environments that could be associated with these two commands. To choose a particular environment for setting a counter or for adding to the current counter, the respective menu button has to be selected. This provides the user with more flexibility. Figure 5 illustrates the various options available for \textit{\texttt{setcounter}} and \textit{\texttt{addtocounter}} commands.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example.pdf}
\caption{The \textit{\texttt{setcounter}} and \textit{\texttt{addtocounter}} Environments.}
\end{figure}

\subsection{Matched Pairing of Commands}

Whenever an environment such as \textit{\texttt{quote}}, \textit{\texttt{itemize}} etc. is selected from the menu options, the corresponding \LaTeX entry inserted into the text window will have the \texttt{\begin{...}} and the \texttt{\end{...}} declarations for that environment. This is performed in order to keep consistency in the command syntax.

Once the environment declarations are inserted into the text window, the cursor is placed in between the two declarations (The \texttt{\begin{...}} and the \texttt{\end{...}} declarations.)
2.2.1 Creating Bibliography

The "Bibliography" command option is chosen to generate a bibliography for the current document. A separate window is created for adding bibliographic items. The user is prompted for the label width for the bibliography. The user could then add the bibliography items in the window. To write the bibliography into the text window, the "Save" menu option is to be selected. Figure 6 enunciates an example bibliography.

2.5 Drawing in EZTeX

We have used the popular drawing program "\texttt{xfig}" for drawing figures. \texttt{xfig} is a public domain software that is freely available for use. The two options provided under the "\texttt{Draw}" menu option invoke \texttt{xfig} when selected. The "\texttt{New}" submenu option prompts for a file name before invoking \texttt{xfig}. If no file name is provided, \texttt{xfig} is invoked without any arguments. The "\texttt{Edit}" command option

Figure 6. An Example Bibliography.
displays all the "fig" files in the current directory and lists the file names in a "ScrollingList Window." To invoke \texttt{xfig}, the user chooses the file name from this list. This invokes \texttt{xfig} with its argument as the selected file name.
CHAPTER III

PROGRAM DESIGN AND METHODOLOGY

In Chapter II we had described the various features offered by our program. In this chapter, we will describe the implementation details for EZT$X. We also discuss the various types of data structures used in EZT$X.

Since EZT$X is based on the X Window System and is built on an OPENLOOK™ GUI platform, we have devoted the next two sections for introducing the reader to the X Window System and the OPENLOOK™ graphical user interface. This is essential for understanding the theory behind the various data structures and system calls used by EZT$X.

3.1 An Introduction to the X Window System

From Apple Macintoshes to Sun SPARCstations and DECstations, built-in graphical capabilities have made graphical user interfaces such as the Macintosh user interface and MS-DOS user interfaces possible. With these capabilities, a user is able to execute commands and run programs using a pointing device such as a mouse, instead of using the keyboard. These graphical interfaces also enable the user to divide the screen into several rectangular regions viz., windows. This makes it possible for the user to run different programs in different windows.

The convenience of using a graphical user interface or GUI is offset by the fact that developing such an interface is an arduous task. In applications such as MS-WINDOWS or the Macintosh user interface, the windowing systems are linked to the underlying hardware and the operating systems of these machines. This causes a potential problem to application developers who would like to write software that runs on many different workstations.

22
The X Window System (or X) was developed in order to create a system that could work on a variety of workstations without any modifications. That is, to achieve what is known as device-independence. This in effect means that the applications written on the X Window system do not depend on the underlying hardware and could be ported to any computer that supports the X Window system.

3.1.1 What is X?

The X Window System is a device-independent windowing system developed by Robert Scheifler, Jim Gettys and their team at the Massachusetts Institute of Technology. It uses a standard set of library routines and a large number of primitive operations to create windows, draw within windows, and handle input from mouse and the keyboard.

X is a combination of several things—the X protocol, X display server, X client applications, and Xlib routines. The X Window System performs its actions based on data (which are commands to X) received through a communication path between the application calling the library routines (Xlib) and the windowing software on the workstation. This implies that the X Window System is capable of displaying, at the workstation's screen, output from an application running on a remote computer that is connected to the workstation through a communication link such as Ethernet or a serial RS-232C connection.

This idea is implemented through a well-defined protocol, called the X protocol, to interpret the data stream sent by the application. X uses a programming model known as the client-server model in which the windowing software running in the graphics workstation is the server performing tasks requested by client applications.
3.1.2 The X Server

The functionality of the X Window System is the X server—the process executing on the workstation and managing the graphics output and the input from the keyboard and mouse. Figure 7 depicts a simplistic view of an X display server and the different modules that read and process X protocol requests from clients over a network connection.

3.1.3 Windows and Events

One of the basic protocols requests the X server handles is to create a window. In X, windows are arranged in a parent-child hierarchy. This results in a tree-like hierarchy of windows created in an application.

While running X applications, everything on the screen appears in windows and each such window is associated with a specific client. Whenever the user does anything with the keyboard or the mouse, it is considered to be an event by the...
X server which sends these events to the client that originally created the window in which these events are generated. For example, if the user presses a key on the keyboard, a *keypress* event is sent to the window which has the *input focus*. A similar type of action takes place when a mouse button is pressed. Another kind of event, an *expose event* is sent to clients if certain things happen to the window such as the window is being overlapped by other window(s) or if the window is moved and configured. It is the responsibility of the client, not the server, for maintaining the appearance of a window.

### 3.1.4 Xlib – The Assembly Language of X

The X Window System is comprised of 300 odd utility routines which are collectively known as the Xlib library. These are primitive functions that enable a programmer to create and manage windows, for drawing in these windows and for managing events. These functions are analogous to the assembly language routines. Programming in X (using Xlib functions) is a tedious process but the advantage of creating an application using Xlib library is that the Xlib library is not system dependent and could be ported to different machines that run X Windows software.

### 3.1.5 X Toolkits – The High-Level Languages of X

A major drawback in creating an application solely based on Xlib functions is the time and effort involved in doing so. For example, for creating a simple window such as a menu button involves using a substantial amount of Xlib calls. To address this problem, the *X Toolkit Intrinsics* (also known as *Xt Intrinsics*) system was developed. These intrinsics are used to create certain basic objects such as menu buttons, popup windows, canvas windows etc. These objects are known as *widgets*. Xt Intrinsics make programming in X windows a lot easier. Xt
Intrinsics is equivalent to a higher-level programming language such as C, PASCAL etc.

3.2 The OPEN LOOK™ Graphical User Interface

The designers of the X Window System designed it in such a way that the programmers using X are free to design any kind of an interface as they pleased. As a result, many graphical user interfaces such as the OPEN LOOK™ and the OSF/Motif interfaces have evolved that are based on the X Window System. X in itself is not a graphical user interface. It merely provides convenient tools (Xlib, Xt Intrinsics etc.) for the programmer to create windows, draw in them and to process the various events such as keyboard and mouse movements that are associated with these windows.

3.2.1 What is a Graphical User Interface?

The appearance and behavior of an application is determined by the application’s user interface. If the user interface makes use of graphical objects such as windows and menus, it is called a graphical user interface (or GUI in short). Currently there are a number of graphical user interfaces available in the market for different computers. For example, the Microsoft Windows is a popular GUI for MS-DOS based personal computers. The two popular graphical user interfaces for X Windows based systems are OSF/Motif and the Openwindows/OPEN LOOK™ system. OPEN LOOK™ is a vendor software developed by SUN Microsystems Inc. and is available on SUN microsystems products. The software that we have developed is based on the OPEN LOOK™ platform.
3.3 Implementation of the EZT\text{$\Pi$}X Interface

As mentioned in the earlier sections, EZT\text{$\Pi$}X is built on an X Window based system. We have used the C programming language for writing the source code for this interface. Several files are used for the design. Each of these files accomplishes a specific function. The data files used in this interface are mainly used for storing the Bitmaps. Bitmap files are used for creating images for representing mathematical and greek symbols. Although we have tried to include a lot of features offered by IN\text{$\Pi$}X, we left out some of those features that are not commonly used. Moreover, since our program is built in a modular fashion, these features could easily be added in the future releases of the program.

In the following sections, we discuss the programming techniques and data structures that are involved in writing the interface. To describe in detail the various data structures and programming tools used in writing this interface would be daunting and unnecessary for the reader. We will only describe the important programming techniques and tools. We will also discuss the various X Toolkits and their functions.

3.3.1 An Introduction to OPEN LOOK$^TM$ Toolkits

The X Toolkit Intrinsics library provides a number of functions for creating Widgets. A Widget is a user-interface component such as a scrollbar, button or a text window. From a programmer's point of view, a widget is a data structure with hidden internal details. Internally, a widget is created using primitive X functions and Xlib function calls. By putting together one or more widgets, one could create an user interface. We have used a number of widgets for our interface design. The advantage of using widgets is that the programmer doesn't need to bother about implementing window management functions. The widget takes care of all that.
3.3.2 The Class Hierarchy of OPEN LOOK Widgets

All OPEN LOOK widgets classes are subclasses of the Core, Composite and Shell widgets of Xt Intrinsics. There are three distinct categories of widgets in OPEN LOOK.

1. **Primitive** – This widget class includes all stand-alone widgets including StaticText widgets, buttons, scrollbars.

2. **Manager** – Includes the widgets that manage the layout of several child widgets. In this category one could find forms, scrolling lists, and checkbox widgets.

3. **Shell** – These widgets are meant for creating the top-level windows of an application.

Following is a list of widgets used in our implementation of EZTeX.

1. **TopLevelShell** widget is the main widget for any application. All the other widgets used in the application are descendants of this widget. Typically all X applications start by opening a connection to the X server. After a connection is established, a top-level window must be created. The OPEN LOOK Intrinsics toolkit provides the Ollnitialize function to take care of these steps.

2. **FooterPanel** widget is a simple manager widget. It is attached to the bottom of a top-level window.

3. **Form** widget is a general purpose widget. Different types of widgets such as controlArea, scrollbar etc could be attached to this widget.

4. **ControlArea** widget is one of the commonly used widgets. It provides a simple way to manage the layout of several child widgets. Menu buttons, textfield and many more widgets could be attached to this widget.

5. **MenuShell** widget is a manager widget. It is used to create pop-up menus. An important property of this widget sub-class is that a MenuShell can be a child of any widget (even primitive widgets.)
6. **CheckBox** widget manages a label and a check box. When selected, the check box has a check mark.

7. **ScrolledWindow** widgets are useful for implementing a scrollable graphics or text area. It can manage a single child that displays text or graphics output. The main window in EZTeX consists of this widget for scrolling text.

8. **ScrollingList** widgets is for displaying a list of items in a scrollable box. We have implemented this widget for our "Draw" menu option.

9. **Exclusives** and **NonExclusives** widgets are used to create a set of rectangular buttons. In Exclusives widget class, only one of the buttons could be active at any given time. There is no such restriction for the NonExclusives widget class.

10. **Text** and **TextField** widgets are used for displaying and writing text. The TextField widget consists of a fixed field which is used for typing text. On the other hand the Text widgets could be used as text editors. We have implemented our text window based on the TextEdit widget which is similar to the Text widget.

11. **MenuButton**, **OblongButton** and **RectButton** are various types of primitive widgets. These could be attached to a control area or a pop-up window to create menus. The buttons are associated with a callback function. When one of these buttons is selected, the callback function associated with that button is invoked.

### 3.3.3 An Overview of Windows and Their Functions

To start with, two windows are created when EZTeX is invoked. One of these windows is labeled as "EZTeX" which is the main window for the interface. The second window labeled "Message Window" is used for displaying warning or error messages while the program is running. Soon after the program is invoked, the **Ollinitialize** library call is invoked. This is an OPEN LOOK library call which
establishes a connection to the X server, creates a top-level window (or main window) and parses the command line for resource specification. After the top-level window has been successfully created, another top-level window is created. This window (labeled "Message Window") is used for displaying warning and error messages from the program. A footer-panel widget is then attached to the main window. To this footer-panel we attach a form widget which was described in the previous section. A control-area widget and a scrollbar widget are attached to the form widget. We then attach seven menu buttons to the control-area. This set of buttons forms our main menu for the program. After creating the main menu, a TextEdit widget is then attached to the form widget. This completes our basic window structure. The text area is used for typing and inserting text. The file "olutil.c" contains a few useful routines for creating menus. The function MakeMenuButton creates the menu buttons. MakeMenuButton is used for creating all the submenus for the menu buttons. For each of the menu buttons thus created, we associate a callback function to it. There are more than 200 menu options available in our package. There are several windows that are created during the execution of the program. They serve various purposes. Following is a list of such windows and a brief description of their functions.

1. Pop-up windows are generated by menu buttons when selected. These are used for performing various tasks. For example, if the "Find" command option (which belongs to the "Edit" menu) is chosen, a pop-up window appears that consists of two textfield widgets and two buttons. The "Find" command option is used for searching text patterns in the text window.

2. Format Window is used for displaying the current format of the document. The document format is a skeletal representation of the document and contains the various environments in the document. The user is given a choice of viewing the format using xdvi, dvipage or pageview programs.
3. \textit{LaTeX} window is created when the current text contents in the text window are processed using \LaTeX. The output generated as a result of running \LaTeX are displayed in this window.

4. \textit{Spell} window is created when the "\textit{Spell}" menu option is chosen, a "TextEdit" window is created that contains the output of running the \textit{spell} program on a text file.

5. \textit{Bibliography} window is used for entering the bibliographic entries.

3.3.4 Software Development and Programming Techniques

In this section, we describe the various programming techniques used for creating this interface. The source code for the EZ\LaTeX package is around 12000 lines. There are a number of data structures involved in creating this interface. To describe the entire program logic is beyond the scope of this thesis. We describe only the important techniques involved in the software development process.

A majority of the command options are used for simple data retrieval from a global database. For example, when the "\textit{Type Size}" command option (which belongs to the "\textit{Document}" menu option) is selected, a submenu is created that contains a list of rectangular buttons. The button labels specify their functions. On selecting one of these buttons, the corresponding \LaTeX code is retrieved from a global database and is inserted at the current cursor location in the text window.

There are a few commands however, that require special attention. Almost all the commands that are involved with the text window need additional processing. The reason behind this additional processing is as follows. The text editor that OPEN LOOK provides does not provide any of the fancy options provided by other text editors (such as searching text patterns) It is the responsibility of the programmer to develop programs to support these features. Fortunately, there are certain OPEN LOOK library calls available that are helpful in writing
such programs. Following is a list of such commands and a brief description of
programming techniques used in creating them.

1. **Math and Greek Symbols** – We have created separate windows that
display the mathematical and greek symbols on the rectangular buttons. When
these buttons are selected, the corresponding \LaTeX declaration is inserted into the
text window. A bitmap was created for each of these symbols. Xlib library calls
were then used to transform these bitmaps into images and then copied onto the
rectangular buttons. Altogether around 178 symbols are available in our package.
Six classes of mathematical symbols are available. These are respectively Binary
Operation symbols, Arrow symbols, Miscellaneous symbols, Relational symbols,
variable-sized symbols and log-like functions.

2. **Displaying Current Document Format** – As mentioned previously, the
"current document format" command option displays the section names, chapters
and other environments. The function "current format" reads the internal text
buffer associated with the text window and parses the lines in the buffer beginning
from the last line till the beginning of the document. While parsing the buffer
the program checks the current line for any of the \LaTeX environments. If a valid
environment is found, it is stored in a database. Once the scan is finished, this
database is written into the format window for displaying the current document
format. Extensive error checking is performed during the scan and any errors are
reported in the message window.

3. **Running \LaTeX on Current Text** – The "Process File" option under
"Properties" menu option is provided for running \LaTeX on the current text con­
tents in the text window. The "Run \LaTeX" command option was implemented
using the \textit{xview} programming interface. When the user selects this command, a
child process is spawned from the parent process which executes a program called
"\texttt{run.latex}" that is implemented using the \textit{xview} programming library. Since there
are no OPEN LOOK functions available for creating \textit{tty} windows (windows that
are capable of running programs), we made use of the xview programming interface to do so. The output generated by \texttt{IAITEX} is sent to this window for viewing. An important feature of our program is that if an error occurs while \texttt{IAITEX} is processing the text, the line number of the first such error is retrieved and after the user closes the \texttt{IAITEX} window, the program places the cursor at the end of the line that contains the error. This makes it easy for editing the text.

4. \textit{Previewing the dvi File} – This option belongs to the "Process File" menu option. When selected, the \texttt{xdvi} program is invoked for the \texttt{dvi} file corresponding to the text in the text window. Before selecting this command option, the text has to be processed using \texttt{IAITEX}. The \texttt{xdvi} program that was originally written by Eric Cooper was modified such that the user can refresh and "reopen" \texttt{xdvi} by selecting a command option from the "Process File" menu. This option was implemented using signal calls. The child process which is the \texttt{xdvi} program, registers a kill signal from the parent (which is \texttt{EZTEX}). Whenever the user selects the "Reopen \texttt{xdvi}" command option, the child process traps the signal sent by the parent which in turn "reopens" the \texttt{dvi} file. If the "Close \texttt{xdvi}" command menu option is chosen, a \texttt{SIGINT} signal is sent to the child process (if it exists) resulting in the termination of the process.

5. \textit{Local and Global Error Processing} – A key feature of our program which is not available in most of the other \texttt{IAITEX} and \texttt{TEX} interfaces is that our program performs error checking at the local and global levels. The current line (The line in the text window where the cursor is placed) is checked for any obvious errors such as incorrect command syntax, invalid environment declaration, invalid arguments to \texttt{IAITEX} commands and if any such errors are found, a warning message is placed in the message window. This type of processing is termed local error processing. On the other hand whenever a new environment is inserted into the text window, the entire document is parsed beginning from the current line going backwards to the beginning of the document. If an inconsistency is found
in the environment declarations or if the "\begin" and the "\end" declarations for any environment are not matched, an error message is placed in the message window. This feature keeps the document consistent at all times and reduces the chances of the document containing any obvious errors before being processed by \LaTeX.

6. **Cursor Placement** – Once an environment or a \LaTeX command is inserted into the text window, the current line in the text window is parsed by the program and the cursor is placed inside the innermost nesting of the argument(s) for this command.
CHAPTER IV

A USER'S MANUAL FOR EZT\TeX

In this chapter, we describe the various features offered by EZT\TeX. In each section, detailed instructions are given for every feature that is explained in this chapter.

4.1 Getting Started

To use EZT\TeX, specific hardware and software are needed. Since the EZT\TeX program was developed in an SunOS/OpenWindows environment, this program runs only in the OpenWindows version 2.0 and version 3.0 environment. It is possible to port this program to other X Windows based systems but to do so, the system specific library calls in the source code need to be modified. To run the EZT\TeX program successfully, follow these instructions.

1. Log on to one of the SUN workstations (OpenWindows must be installed on this machine for running EZT\TeX.)

2. Find out the directory where EZT\TeX program is installed in. Set this directory on your default path. For example, if EZT\TeX is installed in “/usr/export/home/sol2” directory, then an alias for EZT\TeX could be defined as “alias EZTeX /usr/export/home/sol2/EZTeX &”. This command could also be added to the .cshrc file in your main directory. This way the alias does not have to be defined every time you log in.

3. Once the alias is defined, EZT\TeX could be executed by typing EZTeX on the command line. This brings up a set of windows. The first window is labeled “EZTeX” and it is the main window of the program where the user types in the
text and the second one labeled as the "Message Window" receives the error and warning messages from the program.

4.2 The File Menu

The File menu deals with the creating and opening documents. The following subsections provide a brief description of the options available in this menu. Since the text window has an internal text buffer associated with it, we will use the term "text buffer" for the contents of the text window.

4.2.1 Creating a New Document

When the "New" menu option is selected, the current text in the text window is cleared and a pop-up window appears. This window consists of a standard set of declarations which are placed in the preamble of the document (In \LaTeX, a preamble is the set of declarations that occur before the "\begin{document}" command.) Figure 8 illustrates the windows created when the "New" command option is selected.

4.2.2 Loading a Document From the Disk

To load a document from the disk, the "Load" menu option has to be selected. As a result of this, a pop-up window appears that contains two fields labeled "Directory" and "File" respectively. The "Directory" field expects a valid directory path name where the file could be found. The user then types in the desired file name in the text field labeled "File". No action is taken if no file name is specified or if the file specified does not exist on the disk. Figure 9 displays the pop-up window associated with the "Load" menu option.
Figure 8. Diagram Illustrating the Use of “New” Menu Option.
4.2.3 To Clear the Current Text Buffer

The “Clear” command option clears the current text contents in the main window.

4.2.4 Saving the Contents of Text Window

The “Save” command option saves the contents of the text buffer on to a disk file. If the current text buffer has not been saved previously and the “Save” option is chosen, then the user is prompted for a new file and directory name where this file is to be saved. Once the current text buffer has been saved to a new file, if this menu option is chosen again, the text buffer is automatically saved on to the associated disk file.

4.2.5 Saving the Text Buffer Onto a New File

This performs the same action as the “Save” menu option except that it always prompts for a new file name and saves the text buffer onto the new file.
4.2.6 Quitting \textsc{EZTeX}

The "Quit" menu option when selected, closes the connection to the X server which subsequently closes the current application.

4.3 The Edit Menu

This menu consists of four options viz., "Find," "Goto Line," "Select All" and "Spell." The "Find" command is used for find text patterns. The "Goto Line" places the cursor at the line number specified by the user. By selecting the "Select All" menu option, the entire text in the text window is highlighted. The "Spell" command option is useful in running the "spell" program available in UNIX. This menu option when chosen, creates a submenu that has three options. These are respectively labeled as "Run Spell," "Open Spell file" and "Close spell file." The "Run Spell" option expects a source and a destination file name which are respectively the input and the output files for the "spell" program. After executing "spell" on the source file, the output appears in the destination file. This file could be viewed by selecting the "Open spell file" menu option. The destination file is opened in a separate window and contains all the misspelled words found by the "spell" program. The "Close spell file" option closes the window that contains the destination file. Please refer to Figure 10 for a look at the various options available under the "Edit" menu.

4.4 The Properties Menu

This menu is used for setting the various properties associated with the document. The options available under this menu are labeled respectively as "Math," "Symbols," "Auto Save," "Show Current Format" and "Process File." Each of these options contain their own menus and submenus. We provide a brief description of these commands in the next few paragraphs.
40

Figure 10  The Various Features Offered by the “Edit” Menu Option.

4.4.1  The Math Menu

This menu option produces a submenu when selected. The various options under this submenu support commonly used mathematical objects such as “Formula,” “Equation” etc. A list of all these options are produced in Figure 11. By selecting any of these options, the corresponding \LaTeX\ commands are inserted into the text window.

4.4.2  The Symbols Menu

This menu generates about 178 different types of symbols used in \LaTeX. These symbols are generated using bitmaps. This menu has a submenu associated with it. The submenu has two entries viz., the “Math” and the “Greek” options. The “Math” option also has a submenu that lists the different categories of symbols available. Figure 12 lists all these categories. On selecting one of these options, a pop-up window appears that contains all the bitmapped images pertaining to that category. Figures 13, 14, 15, 16, 17, 18 provide a list of all

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
such bitmaps. The "Greek" menu option on the other hand lists all the Greek symbols available in \LaTeX. Refer to Figure 19 for a list of the greek symbols available in \EZLaTeX. If the user wants to create a mathematical or greek symbol in the current document in the text window, he/she has to select one of these symbols and the \LaTeX command for that symbol is automatically inserted into the text window. In other words, the user doesn't have to bother about remembering the \LaTeX command for any of the special symbols.

4.4.3 Automatic File Backup

This option is provided for creating a backup copy of the current text buffer. If due to some unforeseen reasons, a power surge causes the application to crash or due to human error the current file contents are lost, the user should be able to recover the file from its backup copy. This option has to be explicitly chosen by the user to activate the automatic backup mechanism. When selected, this option pops up a window labeled "Timed Backup." This window contains
Possibility and can perform local applications. Each site maintains a local application, which

Figure 12. Categories of Mathematical Symbols Available in EZTeX.

General Terms: Distributed databases, Fragment allocation.

Figure 13. List of "Binary Operation Symbols" Available in EZTeX.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 14. List of "Arrow Symbols" Available in \texttt{EZTeX}.

Figure 15. List of "Miscellaneous Symbols" Menu.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 16. List of "Relation Symbols" Available in EZTeX.

Figure 17. List of "Variable-Sized Symbols" Available in EZTeX.
Figure 18. List of "Log-like-functions" Available in EZ\TeX.

Figure 19. List of "Greek Symbols" Available in EZ\TeX.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
a text field that is used for specifying the time interval between two automatic saves. The default time period is 120 seconds (or two minutes.) Once the “Auto Save” feature is activated, a copy of the current file is created. This file has the same name as the current file with a “" attached to the file name. The original contents of the text buffer are kept in this backup file. The current text buffer is repeatedly backed after the specified interval of time. Figure 20 shows the window pertaining to the “Auto Save” command option.

4.4.4 Displaying the Current Document Structure

This option is useful if the user wants to view the skeletal representation of the document. A skeletal representation of the document implies that only the document subsections, sections and other environments are displayed. The “Show Current Format” command option under the “Properties” menu is to be chosen to view the document format. When this option is selected, a new window is opened that has a menu button labeled “Options” attached to it. The document structure is displayed in the lower part of this window. Six command options are available.
through the "Options" menu. The document structure could be viewed using three different programs viz., "xdvi," "dvipage" and "pageview." These programs are public domain programs and can be obtained free of cost from several ftp sites. Refer to chapter I for a list of these sites. To view the structure using any of these three programs, select "Process File" menu option from the "Properties" menu. Now choose "Run LaTeX" command option from this menu. This opens a window for running \LaTeX\ on the current document format. If any errors occur during the execution of \LaTeX, then correct these errors and rerun \LaTeX\ by selecting "Run LaTeX" again. Once \LaTeX\ is successfully executed, select the "Quit" button on top of the "LaTeX" window to close it. The user could now view the document structure using "dvipage," "pageview" or "xdvi" programs by selecting "dvipage," "pageview" and "xdvi" command options respectively. Figure 3 illustrates the document structure using the "xdvi" option. The "Document Structure" window could be closed by selecting the "Quit" command option under the "Options" menu.

4.4.5 Processing the "Current Document"

While typing the document in the text window, if the user wishes to run \LaTeX\ on the current file, the "Process File" is to be selected. This menu option contains of four command options labeled "Run \LaTeX,", "Open xdvi," "Reopen xdvi" and "Close xdvi" respectively. As implied, the "Run xdvi" option when selected, processes the current file using \LaTeX. As described previously, the output from processing of this file is displayed in a window labeled "latex Window". If any errors are detected during the processing, the cursor is automatically moved to the line in the text window where the first such error had occurred.

The "Open xdvi" option is used for viewing the current document using the \xdvi\ program. When this option is selected, the \xdvi\ program opens a new
4.5 Document Properties

This section describes the options available under the "Document" menu option. Four options are available in this menu. These are respectively labeled as "Document Preamble," "Type Style," "Type Size" and "Splitting Input." Figure 21 displays these four options.
4.5.1 Document Preamble

This option lets the user set the various standard declarations associated with a document's preamble. Figure 8 illustrates these declarations. To insert the desired characteristics for the document preamble into the text window, the "Save" button has to be selected. In doing so, one has to make sure that the cursor is placed before the \begin{document} declaration. (since the preamble of a document occurs before the \begin{document} declaration.)

4.5.2 Selecting Text "Size" and "Style" for the Document

As the labels imply, the "Type Style" and "Type Style" options let the user select the desired type style or size of the document. Upon selecting one of these options, the corresponding \LaTeX code for the chosen option are inserted into the text window at the current cursor location. The various type sizes and styles available in \LaTeX are represented in Figure 21.

4.5.3 Including External Files

\LaTeX provides an option for including external files into the current document. \LaTeX provides three such commands for including files. These are respectively the \texttt{\include}, \texttt{\input} and \texttt{\includeonly} commands. The first two of these commands let the user include a file at the current cursor location. The \texttt{\includeonly} command on the other hand, specifies the file name(s) that are to be excluded from the current file. For a detailed explanation of these commands, refer to the \LaTeX reference guide. \LaTeX offers these commands to the user under the "Splitting Input" menu option. These options are respectively labeled as "Input File," "Include file" and "Includeonly." When these options are selected, a pop-up window appears that lets the user type in the directory and the file name. The "includeonly" lets the user type in multiple file names by separating these
Figure 22. The Three Options Provided by the “Splitting Input” Menu Option. names by commas. If the specified files do not exist, an error message is placed in the message window. The “Input File” and “Include File” do not permit multiple file names to be included. Figure 22 illustrates the “Splitting Input” menu option along with its three command options.

4.6 The Layout Menu

This menu supports most of the commands and declarations used in \LaTeX. There are thirteen command options available under this menu. Some of these options have their own menus and submenus. We describe these options briefly. Figure 23 displays these thirteen options.

4.6.1 Sectioning

Figure 24 shows a list of options available under the “Sectioning” menu option. When each of these options are selected, their corresponding \LaTeX declaration is inserted into the text window.
4.6.2 Sentences

This menu contains eight options, some of which have their own menus and submenus. Figure 25 represents these eight options.

1. Dashes menu option contains three menu items labeled "Intra-word," "Inter-word" and "Punctuation" that respectively. These commands are used for inserting "dashes" in the text buffer.

2. Spacing menu option provides two types of spacings in a \TeX file. These are "Paragraph Mode Spacing" and "Math Mode Spacing" for adding spaces in the \texttt{paragraph} mode and in the \texttt{math mode} respectively. Figure 26 illustrates this feature.

3. Special Characters option is useful in inserting the \LaTeX\ declarations for certain characters on the keyboard that cannot be inserted as such. For example, the characters \{ and \% have to be written as \texttt{\{} and \texttt{\%} respectively in \LaTeX. Refer to Figure 27 for a list of these characters.

4. Emphasis, Unbreakable Text and Footnotes options insert their corresponding \LaTeX\ commands. The Date option inserts the \LaTeX\ declaration for the current date.

5. Logos menu option provides two options viz., "\TeX" and "\LaTeX" for creating logos for \TeX and \LaTeX\ respectively.

4.6.3 Environments

The "Environments" menu option provides six submenus that let the user select an environment. For example, to select the "document" environment, the "Document" option is to be selected. A list of all such environments supported by EZ\TeX\ is provided in Figure 28.
4.6.4 Declarations

A set of declarations available under this option are shown in Figure 29. A declaration in \LaTeX is a command that does not have any arguments.

4.6.5 Defining, Cross-Referencing and Indexing

The "Define" menu option lets the user define new commands and environments. The "Cross References" option lets the user provide labels and references to various objects such as figures, tables etc. Figure 30 shows the two menus corresponding to the "Define and "Cross References" menu options. The "Index" option in the "Layout" menu is useful in creating indices for various objects. These objects are depicted in Figure 30. The options "Table Of Contents," "List Of Figures" and "List Of Tables" when selected, respectively insert the \TeX declarations for these commands into the text window.

4.6.6 Reading Input and Writing Output Interactively

In \LaTeX, the "typein" command lets the user write text into the text window interactively. Similarly the "typeout" command lets the \LaTeX program to write a specified output string to the standard output (the screen). The "Typein" and the "Typeout" commands under the "Layout" menu perform these actions respectively.

4.6.7 Creating Bibliography

The "Create Bibliography" option lets the user create a bibliography for the current document. When this option is selected, A pop-up window appears that requires the user to enter the maximum label-length in the current bibliography. Upon entering the label length and by selecting the "Load" button present on the bottom of this window, a separate text window labeled "Bibliography" window is created that contains three menu options labeled "Add Item," "Save" and "Quit."
The "Add Item" option lets the user add new items. The "Save" button inserts the contents of the bibliography window into the main text window. The "Quit" button naturally, closes the bibliography window. Refer to Figure 6 for an example bibliography.

4.7 Drawing in \textsc{EZT\textit{e}X}

To draw figures, we have made use of the \texttt{xfig} program available freely at public domain sites. For the draw option to work successfully, the \texttt{xfig} program has to be installed on the system. Also before running \textsc{EZT\textit{e}X}, make sure that the path for \texttt{xfig} is properly defined. To define the variable \texttt{xfig}, set an alias for \texttt{xfig} as follows. First find out the absolute path name where the \texttt{xfig} program exists on the disk. let us assume that the path thus found is called path-name. On the command line, type "alias xfig 'path-name.' " To keep this information permanently, edit the \texttt{.cshrc} file in the root directory and insert the above mentioned command into the file.

To invoke \texttt{xfig} from the current menu, two options are provided under the "Draw" menu option. These are respectively labeled as "New" and "Edit." When the New option is selected, \texttt{xfig} is invoked without any arguments (no file name is provided as an argument to \texttt{xfig}.) After using \texttt{xfig}, one could quit \texttt{xfig} and return to the main menu of \textsc{EZT\textit{e}X}. On the other hand if the user wants to edit an \texttt{xfig} file that exists on the disk, the Edit option could be used. Upon selecting this option, a pop-up window appears that contains a list of \texttt{xfig} files found in the current directory. The user could then select the file he/she wants to edit and \texttt{xfig} is invoked with the selected file name as an argument. The list of files appears in the form of a scrolling list. Figure 31 depicts an example directory and a list of \texttt{xfig} files in the directory are displayed in a scrolling window.
4.8 The Format Menu

The format menu lets the user specify the characteristics associated with the text format of the LaTeX document. The three options that belong to this menu are labeled “Page,” “Paragraph” and “numbering” respectively. Each of these options have their own menus and submenus associated with them. A brief description about these commands follows.

4.8.1 Page Format

The “Page” menu option consists of three options labeled “Line Breaking,” “Page Breaking” and “Spacing” respectively. All of these options have menus associated with them. Figure 32 explains displays each menu option and its associated menu. The command options for these menus are self explanatory and are simple declarations. For example, the “Spacing Menu” lets the user specify spacing commands that are available in LaTeX. The “Page Break” menu lets the user set page breaks and other page formatting commands.

4.8.2 Paragraph Format

The “Paragraph” option lets the user set characteristics associated with paragraphs. For example, to draw a line over some text, the “Overline” option is used. The “Box” command has a submenu associated with it which consists of five options that specify the type of box to be created. These box types are shown in Figure 33. Other commands in this menu are self explanatory.

4.8.3 Numbering

The “Numbering” option lets the user set the numbering of various environments. This menu item consists of two options labeled “setcounter” and “addtocounter.” The “setcounter” command sets the counter of an environment variable such as part, chapter etc. When selected, the setcounter option creates
a window which contains a list of environments to choose from. An example of using \texttt{setcounter} is shown in Figure 5. The \texttt{addtocounter} is used for adding the counter value by a specified number. The \texttt{addcounter} works in a similar fashion as the \texttt{setcounter} command.
Figure 23. The Thirteen Options Available Under the "Layout Menu."
Figure 24. Options Available Under the "Sectioning" Menu Option.
Figure 25. The “Sentences” Menu Option and Its Associated Submenu.

Figure 26. The Two Different Modes of Spacing Allowed in EZeTEX.
Figure 27. The Various "Special" Characters Available in Sentences Menu.
Figure 28. The Various Environments Supported by EZ\TeX.
Figure 29. The “ Declarations” Menu and Its Options.

Figure 30. The “Define” and “Cross References” Menus and Their Options.
A distributed database is a collection of data which requires accessing data at several sites using a computer network. Each autonomous processing capability and can perform also participates in the execution of at least one group requires accessing data at several sites using a computer network.

Figure 31. The "Draw" Option in EZT\TeX With an Example Listing of xfig Files.
Figure 32. The "Format" Menu and the Submenus Associated With It.

Figure 33. A List of Options Available Under the "Boxes" Menu Option.
CONCLUSIONS AND DIRECTION OF FUTURE RESEARCH

In this thesis, we have presented the details of a user-friendly visual interface called EZT\LaTeX for the popular typesetting program \LaTeX. This program is built on an X Window interface in a SunOS/OpenWindows environment. We hope that EZ\LaTeX will prove to be a very useful tool for creating \LaTeX documents. It will not only be a great help for beginners but also be beneficial to experienced users.

The EZ\LaTeX program was implemented using the C programming language. Various X Toolkits and Xlib libraries were used to provide the framework for the graphical interface. Since OPEN LOOK does not provide a full fledged text editor, we have provided the basic text editing capabilities. The text editing and updating of the text window is accomplished using text buffers.

This interface provides the user with a majority of \LaTeX commands as defined in the \LaTeX manual. Various menus and pop-up windows are provided that contain these commands in the form of menu buttons. Upon selecting these buttons, the appropriate \LaTeX command is inserted into the text window at the current cursor location.

Several other useful options have been provided such as viewing the dvi file and executing programs such as xfig, xdvi etc. from within EZ\LaTeX. Mathematical and Greek symbols have been implemented as labels for menu buttons. Extensive error checking is performed such that most of these errors are resolved before the current file is processed by \LaTeX.

EZ\LaTeX is a simple program and although the program source code is substantially large, it is not difficult to add new commands and features to it since
the implementation was accomplished using a modular approach. We present a few suggestions for future extensions to our program.

1. *Reconfigurability* – *EZTEX* could be made reconfigurable such that new commands and features can be added by editing a reconfigurable file. This eliminates the need for rewriting the source code for the program.

2. *Synchronizing IATEX and xdvi* – This will allow simultaneous viewing of the IATEX file and the output produced by the *dvi* previewer.

3. *Intelligent Speller* – A new spelling checker could be designed that will interactively check the words in IATEX documents. It should also be able to skip text pertaining to *TEX* and IATEX.
REFERENCES


