Chapter 1

Overview

e6 is a menu-driven assistant for creation and handling of references, particular in the academic sphere. The name e6 or esix means EndNote Sibling In Xlib. It is proposed as an alternative to the *EndNote* commercial product particularly for computer systems which operate under the X Window System. As opposed to *EndNote* which is directed at Word produced documents, e6 in directed towards LaTeX prepared documents which use biblatex to typeset their references. Although *EndNote* and e6 have the same parent of reference handling for document creation, like most siblings they have different lives of their own.

e6 is designed to require minimal software installed on a system to support it. To run e6 and Xx Window server is required. To compile e6 a C compiler with its libraries and standard header files, together with X Window standard header files and the Xlib library, are required. No X Window toolkits are needed.

e6 follows the entry types and associated fields as defined in the *Database Guide* section of the documentation manual which accompanies the biblatex package available from http://www.ctan.org/pkg/biblatex. Alternately http://sourceforge.net/projects/biblatex/files/development contains the latest release of that package. Specifically version 1.4b of that documentation was used in developing e6. Inspecting documentation for later versions of biblatex suggest minimal changes to reference types and fields defined for those types has occurred.

Only standard reference types as defined in biblatex are currently implemented in e6, and not all of those. Provision has been made for extending those types when agreement is reach to what fields should be linked to each of those types. Those extensions are shown in the $Citating \longrightarrow Extensions$ menu.

1.1 Background

The purposes for writing e6 were:

- support for using BibLaTeX referencing, and

- a software project to demonstrate X Window programming via xlib.

BibLaTeX is more complex than BibTeX. The 14 or so reference types in BibTeX are now replace by nearly double that number. The 25 fields used to compose those BibTeX reference types number 88 in BibLaTeX. So knowing which fields belong in which reference type and which of those fields are required as opposed to optional in each of those reference types has become more difficult. There are situations when an overlooked field in preparing a reference cannot be overcome and so assistance in getting everything right when the reference is first prepared is essential. A result of such increased detail in a reference is a better citation in the final document through the styles which use BibLaTeXreference format. Despite such difficulties, there appears to be an increasing shift in the research community to interest in using, if not currently using, BibLaTeX inplace of BibTeX. Providing an assistant in such progression is warranted.

There is an argument that what is implemented in `e6` would be better done as programs/macros in emacs or as a web php/javascript application. Having `e6` as a separate and specific application follows the Unix principle of doing one thing, and doing it well.

Although widely used, emacs is not the standard editor on a Unix system. Wether emacs's program execution platform could provide the visual interface of `e6` is an open question. But the programmability needed to implement `e6` logic and display is not available under vi and it's clones. However, the output of `e6` is a text file which can be subsequently manipulated by emacs or vi after a reference has been formulated by `e6`. The one program `e6` with the one interface is used as the step before the editor of choice.

A web based application imparts the uneasy lingering thought that the references might be available to *the world*. This is not the intent. Your references are yours, and yours alone; for you to choose how they should be made available. So a program on my computer with my references appears a good relationship.

Xlib together with the C language are used to create `e6`. The conventional logic is the use of the Xlib library alone to implement an X Window visual interface is too time consumming. This is true if the program is to be used a few times and thus the time involved in writing it should be minimized. It is anticipated `e6` will be used many times and the additional time required to write it as opposed to using a toolkit is justified. One difficulty in using Xlib alone is learning how this programming is done. Once mastered, opportunities appear. Because Xlib is basic, the predefined manners of behaviour of a toolkit can be ignored enabling the desired behaviour more appropriate to `e6` to be implemented. Also, Xlib is a standard component of any X Window installation in contrast to toolkits which are can be optional extracts: They may be present on the system or they might not. The C language is standard on a Unix system which enables programming. As a result `e6` should be fast to executed and portable across platforms.

The creation of `e6` using Xlib explored the difficulties in writing graphical software using such a low-level library. Something more than a couple of pages of code was needed for such an exploration. With fewer contraints impoosed by the software being used more work is required but more choices become available in the creative

process. It is believed e6 at least shows good outcomes can be achieved if software engineering principles are followed.

It is hoped the availability of e6 with its design and implementation characteristics will assist the use of BibLaTeX as a means of writing and their subsequent use – its limitations not with-standing.

1.2 Current state of implementation

Table 1.1: Reference types currently implemented in e6

Implemented	Not present currently
Article Book Book in Book supplementary book Booklet Collection supplementary Collection in Collection in Proceedings in Reference Manual Miscellaneous Online Periodical supplementary Periodical Proceedings Reference Report Thesis in Book multi-volume Book multi-volume Collection multi-volume Proceedings Patent Unpublished	Set
Artwork Audio Commentary Image Legislation Legal Letter Movie Music Performance Review Software Standard Video	Bibnote

From the outset, e6 was meant to evolve with more reference types added as the need arose. Table 1.1 shows the reference types currently implemented. Space

in the menu layout of e6 has been provided to accommodate BibLaTeX reference types currently not implemented.

Table 1.1 also shows additional reference types which have been allowed for in the layout of e6 which are not defined in BibLaTex. Fields for inclusion in such reference types need to be agreed upon and implemented in BibLaTex before being implemented in e6.

Below the horizontal line in Table 1.1 are the BibLaTeX Unsupported types. These are supported in newer reference style processing packages such as oxyear by Alex Ball and are provided in e6. In accord with oxyear, the Commentary reference type has been replace by the Jurisdisction reference type. The data field contents in these Unsuported types are more likely to change than the data fields of the standard types as expereince with them gathers.

In operating e6, moving the pointer over a reference type will show it as available or not. A red background for the menu element indicate the reference type is implemented. A menu element with a pink background indicates the associated reference type is not currently implemented.

1.3 Compatibility

The version of e6 to which this document applies has been compiled and run on 32 bit and 64 bit Intel processors running Linux. Since only standard C and X Window xlib library calls are used in this program, porting to other environments with such support should be possible.

All package development was performed on a laptop with a 32 bit Intel processor running Ubuntu Linux.

1.4 License

Chapter 2

Operation

This chapter is the user manual. The layout of the graphical services with respect to control of `xm32k`, the menus, and keyboard short cuts of those menu items are discussed.

2.1 Starting `e6`

The standard way of starting the execution of `e6` is from the command line. The standard command:

```
e6 &
```

starts `e6` executing and produces the opening screen shown in Figure 2.1a: The yellow splash screen. The root of the menu system used to control `e6` runs down the top left-hand corner of that screen. Generally `e6` is run as a background process which is produced by the `&` at the end of the above standard command.

The command line arguments listed in Table 2.1 can be used when starting `e6`. If such argumenets are not used, the menu system of `e6` can be used to supply such data.

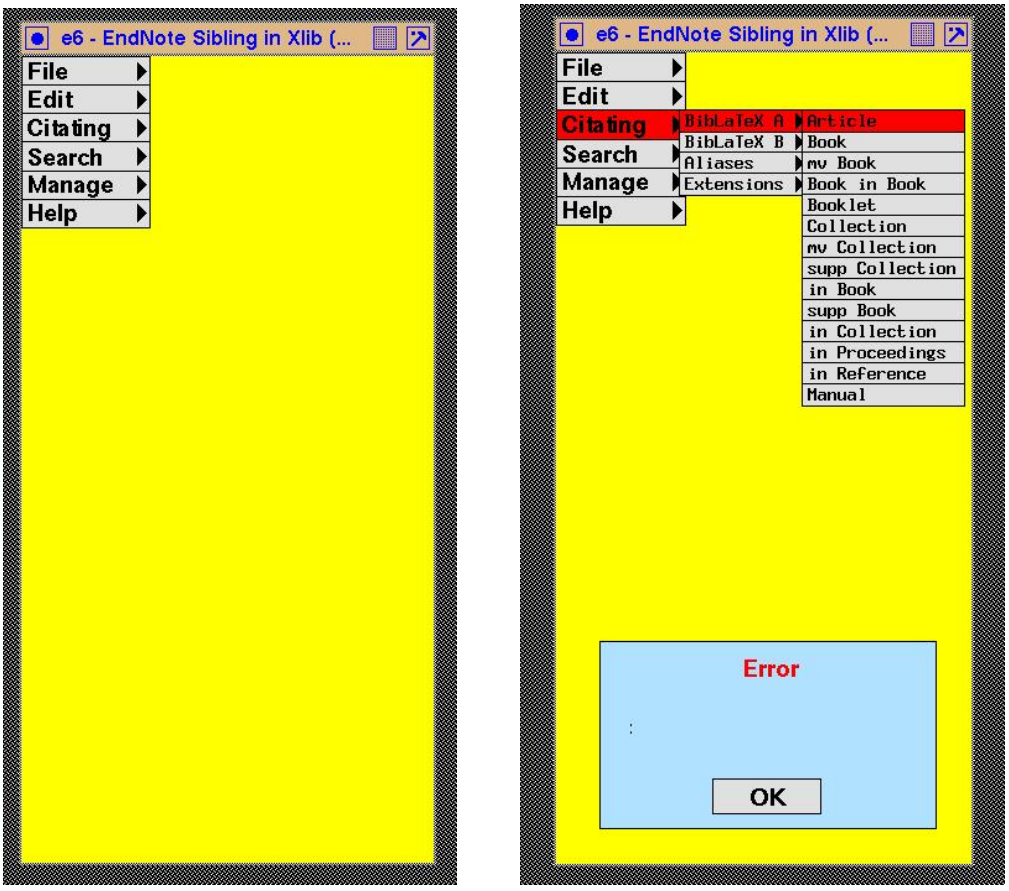
Table 2.1: Available command line arguments

Argument	Meaning
-a ”string”	references created to be appended to file ”string”
-n ”string”	create new file with name ”string” to hold references
-r ”string”	extract references from file ’string”
-v	print version number of this program
-h	print list of available command line arguments

2.2 Menu system

The root menu system of `e6` is located on the top left-hand corner of the display. A right-hand facing arrow head on the right-hand side of a menu entry indicates an-

other menu is attached to that menu entry. An ellipsis (. . .) on the right-hand side of a menu item's label indicates an dialog window will follow requiring keyboard entry.



(a) Default (b) No file

Figure 2.1: Starting `e6` without command line arguments

When the mouse pointer is moved over a menu entry that entry is highlighted. The menu array which leads from that item also appears on the screen. If the mouse pointer is moved into that follow-on array, the menu item which led to it remains highlighted to show the path taken.

Each menu item is colour coded. An unselected menu item is coloured light grey. When the mouse pointer enters a menu item the colour of that menu label changes to red. If a menu follows on from the mouse pointer entering a menu label, that label remains coloured red. A pink colouring of a menu item, when the mouse pointer enters it, indicates no action is assigned to that item in this version of `e6`.

Menus are nested to a maximum depth of two. A selection is at the bottom of each such tree. The selction is made by clicking the left-hand mouse button over the menu item.

2.2.1 Menu composition

The `File` menu contains executions points which control the environment in which `e6` works. Files for input of reference citations into `e6` for editing or searching are linked through the `Read` item. Files for receiving reference information created through `e6` are linked by the `New` or `Append` item. The `New` item starts are new file for storing references, while the `Append` item stores the reference created in the current `e6` run at the end of a currently existing file. Execution of `e6` is terminated through the `Quit` menu item.

The `Citating` menu is divided into four. `BibLaTeX A` and `BibLaTeX B` divided the number of standard reference types into lists of more usable proportions. The `Aliases` menu is a collection of backward compatible reference types and specialized types which are linked to standard BibLaTeX reference types. The `Extensions` menu contains pointers to non-BibLaTeX reference types.

A linked file can be searched using the `Search` menu. Different manners of performing such searching are supported through menu items here. The `Show all` menu item shows the keyword, year, author or editor, and title of all references in the file linked for reading.

The `Edit`, `Manage` and `Help` menus currently have no actions in this version of `e6`.

2.3 Files required

Most operations in `e6` require a file to be opened. This file, or files, can be supplied by the user through a command line argument when invoking `e6`, or by the `e6` menu system. One file to read and another to write can be open at one time. These read and write files must be different, i.e. not the same file for both read and write.

If the operation is to create or edit a reference or references, then a file needs to be opened for writing. Table 2.1 shows this filename can be supplied using the `-a` or `-n` command line options. Alternately the `e6` menu system leading from the `File` item can be used. If such a file is not opened first an error is displayed as indicated in Figure 2.1b.

2.4 Error system

When an error is detected, usually as a result of information not being supplied to `e6` in the order it expects, an error window appears. This error window is has a sea-green background colour with the word `Error` written at it's top. All such error windows appear in the bottom left-hand corner of the `e6` window.

The error window consists to two parts. Across the centre of this error window a message describing the error is given. An `OK` button is provided at the foot of the

window. Figure 2.1b shows such an error display when the user attempts to create an article reference before a file to store such a reference has been opened.

The second part prevents any further actions to be taken by `e6` until this error is acknowledged. Acknowledgement is done by clicking the left-hand mouse button over the `OK` button. This acknowledgement removes the error window.

2.4.1 Error messages and their meanings

The content of error messages displayed by `e6` in the error window, together with a little guidance to their meaning is:

Message	Meaning
First open an output file	Need to open a file to store reference
Save or Cancel before quiting	Save or cancel the data entered
Entries present, so need over-ride	How should the current data be handled?
Close the current output file	Close the current reference recording file
An output file is already open	A file to record a reference is already open
Requested file already exists	File of that name already exists
Could not create requested file	No more storage is available for recording
An input file is already open	Reference recording file is already attached
Requested file does not exist	Requested reference file not found
No reference storage is open	No storage for reference data has been set
Output storage will not close	Computer will not accept the reference file
No reference input is open	Give the reference file to search
Input storage will not close	Computer will not close reference input file
First open a file to read	Must give name of reference containing file
First open an output file	Must give name of file to record references
A file for storage is required	Must give name of file to record references
An entry in 'Keywords' is suggested	Keywords data entry is required
Year/Data field entry required	A Year/Data data entry is required
Title field entry required	A Title data entry is required
Author entry required	An Author data entry is required
Journal Title entry required	A Journal Title data entry is required
Author/Editor entry required	An Author/Editor data entry is required
URL entry required	A URL data entry is required
Institution entry required	An Institution data entry is required
Type entry required	A Type data entry is required
Editor entry required	Editor data entry is required
Book Title entry required	A Book Title data entry is required
Number entry required	A Number data entry is required

2.5 Data editing

All data input, and editing of the data entered, is performed through the keyboard. Editing is to correct detected errors when preparing fields of a reference type.

This editing capacity is restricted in its functionality. But further editing can be performed subsequently using a standard text editor.

2.5.1 All fields

The up and down arrow keys are used to move the input cusor to the field above or below, respectively, of the current field.

The `Annotation` is an exception to such movement. The arrow keys can be used to enter the `Annotation` field, but not to exit it. In this field, the up and down arrow keys have the purpose of moving between lines of text within this field. However, this behaviour has not as been implemented in this version of `e6`.

The left and right arrow key are used to move the cursor along a line of text to the position for entering characters within the existing line, or from where characters are to be deleted.

The `Back space` key deletes the character to the immediate left of the cursor's current position.

All fields (except the `Annotation` field) are single line fields.

2.5.2 Note field

The `Note` field is implemented. Multi-line notes are supported. The `e6` implementation of the `Note` field of `BibLaTeX` has a maximum capacity of 21 lines each of 119 characters.

The `Page Up` keyboard key is used to get out of the `Note` field and go to the field immediately above it on the screen. The `Page Down` keyboard key is used to get out of the `Note` field and go to the field immediately following it on the screen. The up and down arrow keys do not perform those functions.

A new line is created by the `Enter` key. The cursor then moves to this new line. Once the cursor moves from a line, no further editing of that line can occur.

A line of text cannot be deleted.

Editing of this field is limited and needs further work.

2.6 Reference data input

Only one reference type can be operated on at a time.

The data fields of each reference type are broken into two windows. The first window contains the primary, or main, data fields. The second, or minor, window contains data fields which are considered less important. No required data fields, i.e. those with a red boarder (see below), appear in a minor window.

The base `e6` screen expands and contracts to accommodate each reference type screen which very in size.

Each reference type input is broken between two window; primary and secondary. No required data fields appear on a secondary window.

2.6.1 Parts of the reference type screen

Each reference type display is composed of two windows, one for primary data and the other, when called, containing data fields lesser in inportance. Related data fields on each window are collect together in groups. Each group is separated from its neighbour by a small vertical space on the display. Not all reference types have the same groups, but when they occur their order on the screen, and the data fields they contain, are the same between reference type windows.

Each reference type display has require data and optional data. Most required data fields are at the top of each reference type screen. The exception is the `Keywords` data field which although required does not fit logically with the other required data fields. These required fields also have a red border. By contrast, optional data have a blue border.

An exception to the required fields being at the top of a reference type screen is the `keyword` field. This field identifies a each reference and is required. It is logically related to the `sortkey` and `xref` fields, both of which are optional. These three fields are found offset from the top on the reference type screen.

The same reference data field in each reference type screen is of the same length.

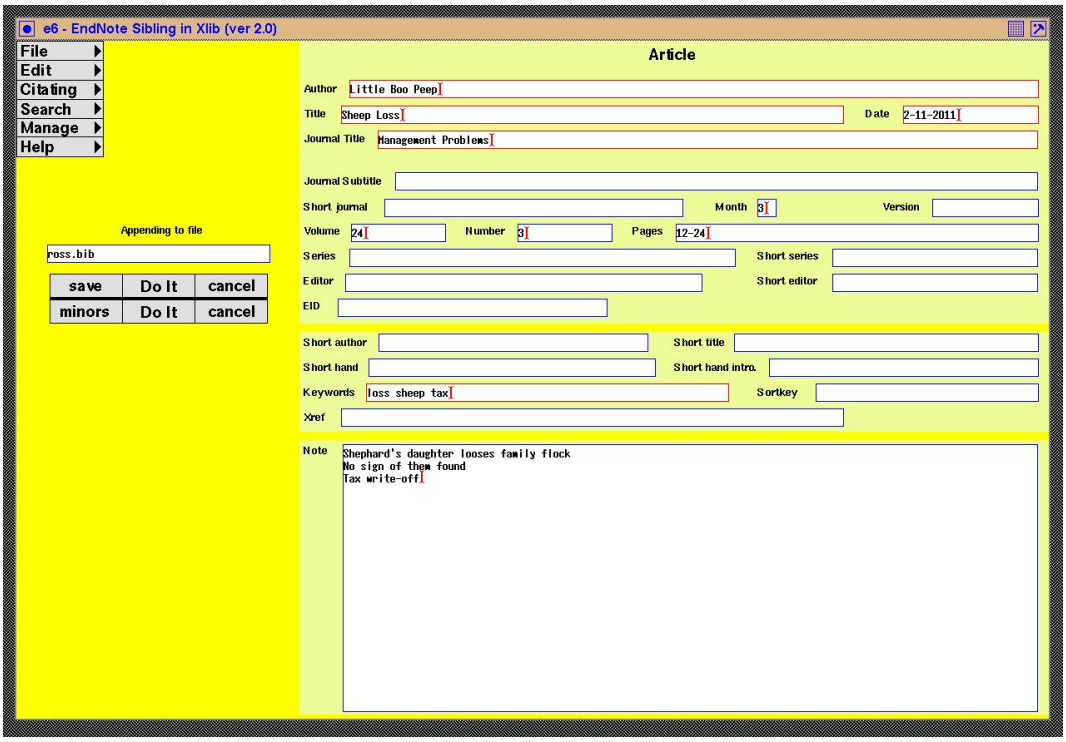


Figure 2.2: Example of primary part of a reference creation

All reference data fields on a screen are linked to their neighbour above and below. The reference data field at the top of a reference type screen is linked to the reference data field at the bottom of the type, and also the converse. An Up-arrow character moves the input cursor to the reference data field above the current. A Down-arrow character moves the input cursor to the reference data field below the current.

Figure 2.2 shows `e6` being used to create an article reference type. Notice:

- the red borders around the required fields as specified by `BibLaTeX`;
- the `Keywords` required data field is not with the other required fields;
- grouping of data fields;

Making the `Keywords` field of `BibLaTeX` a reuired field in `e6` is aimed at assisting search and cataloging multiple references.

The Note reference field appears on all reference type screens. It's purpose is to record notes about the reference which are not printed when the reference is cited. This field is of multiple lines, of which only the first 21 are shown in the Note field on screen. Each reference type has the same size Note field. The maximum number of characters in this field is set to 119 (the value of the NO constant set in the `e6data.h` file of the source code). In such count, each Return (Enter) character is stored as a character.

2.6.2 Action buttons

When a reference type is selected, actions buttons appear on the left of the screen below the field showing the file in which this reference is ultimately to be stored. Those buttons are removed when the reference type screen is removed.

These buttons are grouped into two parts as shown in Figure 2.2. The top row consisting of `save, Do It`, and `cancel` buttons control the primary window of the reference type. The `save` button stores the entered reference information into the file previously opened to receive that data. The `cancel` button removes the current reference type screen. Clicking the `cancel` button after input to a reference field will result in an error. This protection can be overcome by clicking on the `Do It` button, then on the `cancel` button.

The bottom row consisting of `minor`, `Do It`, and `cancel` buttons control the minor window of the reference type. Clicking the `miror` button displays a window containing the secondary data fields for the primary reference type. Then activated this window attaches itself to the mouse pointer as a wireframe. This enables the location of this window to be positioned using the mouse. When the location of the window is decided, clicking the left mouse button displays the contents of this secondary data window. Figure 2.3 show such a window for the article primary reference type. The `Do It` and `cancel` buttons in the bottom row have the same behaviour on this secondary window as their counterparts on the top row have on the primary reference window.

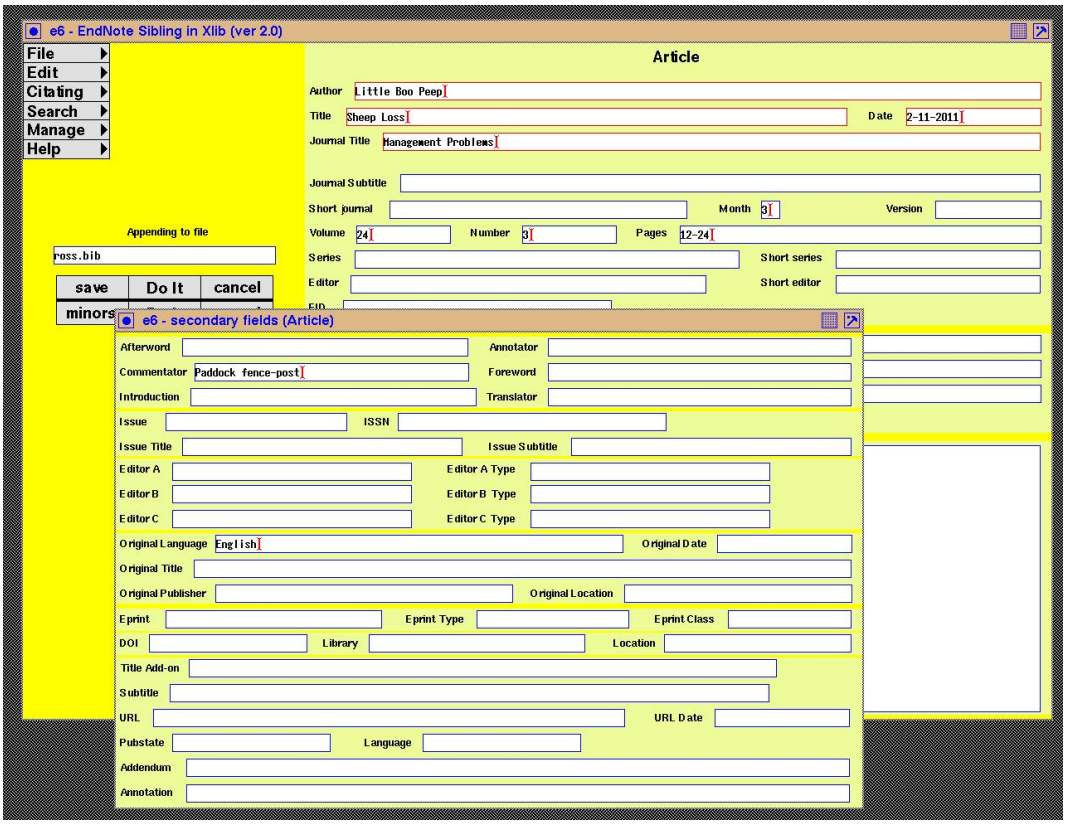


Figure 2.3: Creating the minor part of an example reference

2.7 Special keyboard entry keys

The position of the placement of the next character is shown by the input cursor. The following keyboard characters have special significance when performing input in e6:

Up-arrow	move input cursor to the reference field above the current
Down-arrow	move input cursor to the reference field below the current
Left-arrow	move cursor to the left
Right-arrow	move cursor to the right
Back-space	delete the character to the left of the cursor

The next reference field to receive keyboard data can be set by positioning the mouse pointer over the field then clicking the left-hand mouse button. If the mouse pointer is inside a line of text in the field, the character input will proceed from that point. This point is indicated by the input cursor.

From the point of view of the data fields, the primary and secondary data windows are separate. The above movement keys circulate through the data fields contained in that one window. The mouse pointer needs to be positioned above a data field in the other window for the data fields of that window to be accessed.

2.8 Searching a biblography file

Searching the contents of a file opened for reading is started from the `Search` menu item. Currently there are five catagories of search implemented:

- show all references in the biology file,
- by author of a reference,
- by reference type,
- by date sppecified in the reference, and
- by the occurrence of a word appearing in a reference.

Figure 2.4 shows an example of the display produced when asked to show all references present.

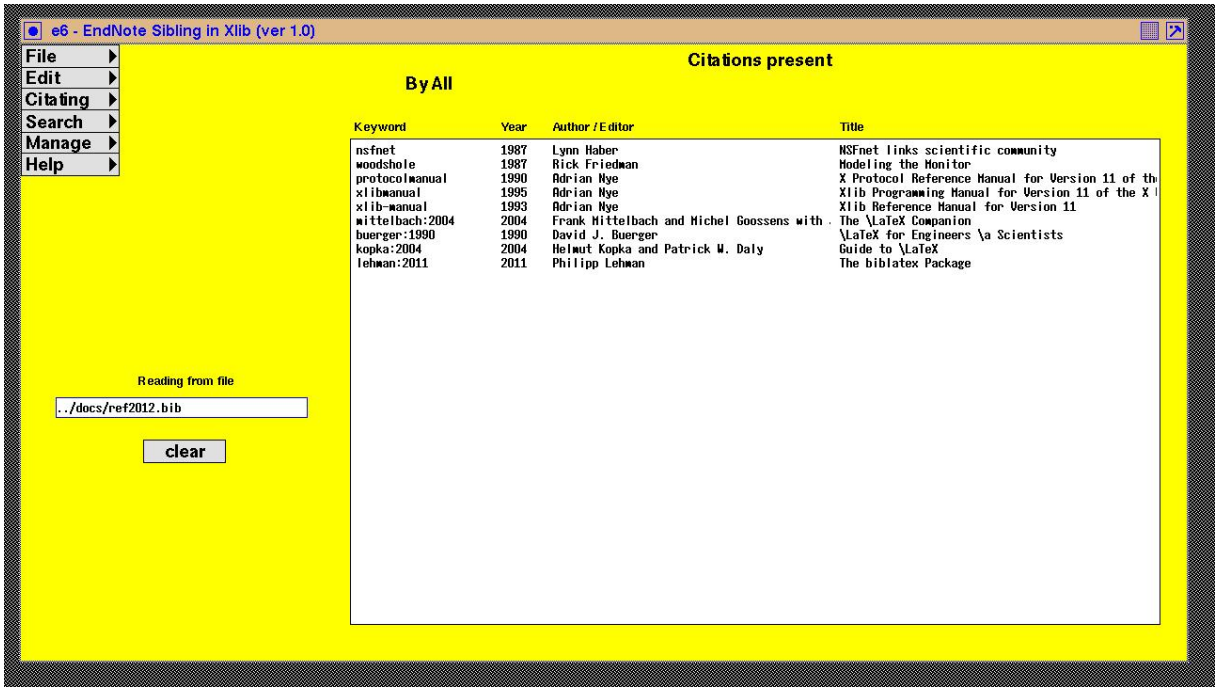


Figure 2.4: Listing the bibliograpphy content

Once `author, type, date`, or `word` search has been selected, a window item for input of the appropriate string will appear at the top of the screen. The search is performed matching that string once the keyboard `Enter` key is pressed.

The `Clear` button below the window item showing the filename being read is used to clear/cancel the current reference search.

Because scrolling has not been implemented in `e6`, approximately only the first 30 matches of a search criteria are shown.

Chapter 3

Software

This chapter records how the software of `e6` is designed to function and how that design is implemented.

The latest version of `e6`, of which this manual is a part, is available from `http://www.yenolam.org/software/e6`.

3.1 Design

A text file contains the resulting reference. Such a file is laid out as direct input to `biblatex`. Because of it's text type, standard utilities such as grep, sed, and text editors can be used on it if the services provided by `e6` are inadequate.

Each input screen is designed to be seen in its entirety without scrolling. This imposes the need for a minimum 1280x1024 pixel screen.

The number of required and optional fields in some reference types (such as in the `article`) dictates the height of the input screen.

3.2 Implementation

3.2.1 Menus constructed from pixmaps

All menus are constructed from labels in the form of bitmaps. Therefore changing their size by scaling will destroy their good looks.

The characters were chosen to be easily legible and easy on the eye of the user.

The labels of the primary menu are chosen to be larger than subsequent menu llabelsThe characters were chosen to be easily legible.

The labels of the primary menu are chosen to be larger than subsequent menu labels.

Primary menu labels have a standard 110x17 pixel dimension. The characters are built from the Helvetica bold 1875 character library.

All other menu labels have a standard 96x6 pixel dimension. The characters are built from the Helvetica bold 1275 character library.

3.2.2 Reference type displays

Table 3.1 and Table 3.2 together show the fields used in the first 11 reference types. Table 3.3 and Table 3.4 show the fields used in the remaining 10 reference types taken from the BibLaTeX documentation. In those tables, an `r` is used to indicate the field is required in a reference type. An `o` shows fields which are optional. An `i` indicates a field is not specified as an optional field in the BibLaTeX but has been inserted as such in `e6`. Such fields appear logical for inclusion and also simplify the implementation of the reference type screens. All fields are held in the source code in the array `dataStore[]`. In the tables the number in the column next to the name of each field is the index in that array coorresponding to the field.

Fields are grouped together in windows stored in the source array `moduleStore[]`. Each of those combination has a name which is show in the second from the right column of the tables. The index in the `moduleStore[]` array corresponding to each combination is given in the far right-hand column of each table. Some field combinations have no name. These field combinations vary in content between reference types and must be assembled as required when the reference type is called.

The reference data fields on each reference type screen are linked in a double-link list through the `struct data` structure defined in `e6x11.h`.

The reference data fields vary in length to correspond to their storage arrays. The same storage array is used to store the contents of a given reference data field for each reference type.

3.2.3 Execution

Upon execution start, `e6` creates all the windows corresponding to menu items and modules containing collection of reference fields which will form the various reference type availables. A separate window for each background colouring of a menu item is created. The windows which appear on the display during the execution of `e6` are assembled from those created parts as needed.

3.2.4 Code distribution between files

The C files which implement `e6` are contained in the `src/` directory. Each reference type is contained in its own source file. The `main()` is contained in the `main.c` file. Functions to handle keyboard input are contained in the `texting.c` file. Functions for error handling are contained in file `error.c`. The function calls

Table 3.1: Part A: Formulation of reference types in e6

		Article	Book	mv Book	in Book	Booklet	Collection	mv Collection	in Collection	Manual	Miscellaneous	in Reference		
Currently in e6		+	+	+	+	+	+	+	+	+	+	+		
author	2	r	r	r	r				r			r		
author/editor	3					r				r	r			
title	1	r	r	r	r	r	r	r	r	r	r	r		
year/date	0	r	r	r	r	r	r	r	r	r	r	r	baseW	
institution	4													
type	61					o				o	o			
edition	62		o	o	o		o	o	o	o		o		
pubstate	23	o	o	o	o	o	o	o	o	o	o	o		
addendum	24	o	o	o	o	o	o	o	o	o	o	o		
language	25	o	o	o	o	o	o	o	o	o	o	o		
note	26	o	o	o	o	o	o	o	o	o	o	o	largeO	0
url	27	o	o	o	o	o	o	o	o	o	o	o		
urldate	28	o	o	o	o	o	o	o	o	o	o	o		
titleaddon	29	o	o	o	o	o	o	o	o	o	o	o		
subtitle	30	o	o	o	o	o	o	o	o	o	o	o		
doi	31	o	o	o	o	o	o	o	o	o	o	o		
library	32	i	i	i	i	i	i	i	i	i	i	i	largeO1	7
location	33	i	o	o	o	o	o	o	o	o	o	o		
eprint	34	o	o	o	o	o	o	o	o	o	o	o		
eprintclass	35	o	o	o	o	o	o	o	o	o	o	o	eprintO	1
eprinttype	36	o	o	o	o	o	o	o	o	o	o	o		
afterword	63	i	o	o	o		o	o	o			o		
annotator	66	o	o	o	o		o	o	o			o		
foreword	64	i	o	o	o		o	o	o			o	wordsO	9
commentator	74	o	o	o	o		o	o	o			o		
introduction	82	i	o	o	o		o	o	o			o		
annotation	65													
authortype	67													
bookauthor	68				o									
bookpagination	69													
booksubtitle	70				o				o			o	baseW	
booktitle	71				r				r			r		
booktitleaddon	72				o				o			o		
chapter	73		o		o	o	o		o	o		o		
editor	16	o	o	o	o		r	r	r			r		
editora	10	o	o	o	o		o	o	o			o		
editorb	11	o	o	o	o		o	o	o			o		
editorc	12	o	o	o	o		o	o	o			o	editorabcO	2
editoratype	13	i	i	i	i		i	i	i			i		
editorbtype	14	i	i	i	i		i	i	i			i		
editorctype	15	i	i	i	i		i	i	i			i		

Table 3.2: Part B: Formulation of reference types in e6

		Article	Book	mv Book	in Book	Booklet	Collection	mv Collection	in Collection	Manual	Miscellaneous	in Reference		
Currently in e6		+	+	+	+	+	+	+	+	+	+	+		
eid	75	o											baseW	
venue	58													
eventdate	76												eventO	10
eventtitle	77													
file	78													
holder	79												baseW	
howpublished	80					o					o			
indextitle	81													
isbn	20		o	o	o		o	o	o	o		o		
number	18	o	o	o	o		o	o	o	o		o		
pages	21	o	o		o	o	o		o	o		o		
part	55		o		o		o		o			o		
publisher	56		o	o	o		o	o	o	o		o	baseW	
series	22	o	o	o	o		o	o	o	o		o		
volume	19	o	o		o		o		o			o		
volumes	60		o	o	o		o	o	o			o		
isan	83													
ismn	84												baseW	
isrn	85													
issn	37	o												
issue	38	o											issO	4
issuesubtitle	39	o												
issuetitle	40	o												
journalsubtitle	6	o											baseW	
journaltitle	5	r												
mainsubtitle	41		o		o		o		o			o		
maintitle	42		o		o		o		o			o	mainO	5
maintitleaddon	43		o		o		o		o			o		
month	44	o									o			
nameaddon	45												baseW	
organization	46									o	o			
origdate	47	i	i	i	i		i	i	i			i		
origlanguage	48	o	o	o	o		o	o	o			o		
origlocation	49	i	i	i	i		i	i	i			i	origO	6
origpublisher	50	i	i	i	i		i	i	i			i		
origtitle	51	i	i	i	i		i	i	i			i		
translator	52	o	o	o	o		o	o	o			o		
pagetotal	53		o	o		o	o	o		o				
pagination	54												baseW	
reprinttitle	57													
version	59	o								o	o			
keywords	86	i	i	i	i	i	i	i	i	i	i	i		
sortkey	87	i	i	i	i	i	i	i	i	i	i	i	keysO	8
xref	88	i	i	i	i	i	i	i	i	i	i	i		

Table 3.3: Part C: Formulation of reference types in e6

		Online	Patent	Periodical	Proceeedings	mv Proceedings	in Proceedings	Report	Thesis	Unpublished	Reference		
Currently in e6		+	+	+	+	+	+	+	+	+	+		
author	2		r				r	r	r	r			
author/editor	3	r											
title	1	r	r	r	r	r	r	r	r	r	r		
year/date	0	r	r	r	r	r	r	r	r	r	r	baseW	
institution	4							r	r				
type	61		o					r	r				
edition	62										o		
pubstate	23	o	o	o	o	o	o	o	o	o	o		
addendum	24	o	o	o	o	o	o	o	o	o	o		
language	25	o	i	o	o	o	o	o	o	o	o		
note	26	o	o	o	o	o	o	o	o	o	o	largeO	0
url	27	r	o	o	o	o	o	o	o	o	o		
urldate	28	o	o	o	o	o	o	o	o	o	o		
titleaddon	29	o	o	i	o	o	o	o	o	o	o		
subtitle	30	o	o	o	o	o	o	o	o	o	o		
doi	31		o	o	o	o	o	o	o		o		
library	32		i	i	i	i	i	i	i	i	i	largeO1	7
location	33		o	i	o	o	o	o	o	o	o		
eprint	34		o	o	o	o	o	o	o		o		
eprintclass	35		o	o	o	o	o	o	o		o	eprintO	1
eprinttype	36		o	o	o	o	o	o	o		o		
afterword	63										o		
annotator	66										o		
foreword	64										o	wordsO	9
commentator	74										o		
introduction	82										o		
annotation	65												
authortype	67												
bookauthor	68												
bookpagination	69												
booksubtitle	70						o					baseW	
booktitle	71						r						
booktitleaddon	72						o						
chapter	73				o		o	o	o		o		
editor	16			r	r	r	r				r		
editora	10			o							o		
editorb	11			o							o		
editorc	12			o							o	editorabcO	2
editoratype	13			i							i		
editorbtype	14			i							i		
editorctype	15			i							i		

Table 3.4: Part D: Formulation of reference types in e6

		Online	Patent	Periodical	Proceeedings	mv Proceedings	in Proceedings	Report	Thesis	Unpublished	Reference		
Currently in e6		+	+	+	+	+	+	+	+	+	+		
eid	75											baseW	
venue	58				o	o	o						
eventdate	76				o	o	o					eventO	10
eventtitle	77				o	o	o						
file	78												
holder	79		o									baseW	
howpublished	80									o			
indextitle	81												
isbn	20				o	o	o		o	o	o		
number	18		r	o	o	o	o	o			o		
pages	21				o		o	o	o		o		
part	55				o		o				o		
publisher	56				o	o	o				o	baseW	
series	22			o	o	o	o				o		
volume	19			o	o		o				o		
volumes	60				o	o	o				o		
isan	83												
ismn	84											baseW	
isrn	85							o					
issn	37			o									
issue	38			o								issO	4
issuesubtitle	39			o									
issuetitle	40			o									
journalsubtitle	6											baseW	
journaltitle	5												
mainsubtitle	41				o		o				o		
maintitle	42				o		o				o	mainO	5
maintitleaddon	43				o		o				o		
month	44	o	o	o	o	o	o	o	o	o			
nameaddon	45											baseW	
organization	46	o			o	o	o						
origdate	47										i		
origlanguage	48										o		
origlocation	49										i	origO	6
origpublisher	50										i		
origtitle	51										i		
translator	52										o		
pagetotal	53				o	o		o	o		o		
pagination	54											baseW	
reprinttitle	57												
version	59	o	o					o					
keywords	86	i	i	i	i	i	i	i	i	i	i		
sortkey	87	i	i	i	i	i	i	i	i	i	i	keysO	8
xref	88	i	i	i	i	i	i	i	i	i	i		

which create the windows used by all other functions are in the files `foundations.c` and `modules.c`. Other files provide specific support functions.

There are two header files. The file `e6x11.h` contains the definitions of variable used in common between functions, particularly those used with calls to the `xlib` library. The file `e6data` defines the arrays which contain the reference fields.

All `xbm` files store a single pixmap.

3.2.5 Package folder layout

The package content is distributed over five subdirectories as shown in Table 3.5. The root directory contains the `README` which gives an overall description of `e6`, and the `Changelog` file which gives some insight to the changes introduced into each version of `e6`.

Table 3.5: Subdirectory content of the `e6` package

Subdirectory	Contents
cbg	source code to build `cbg` to create text pixmap glyphs
composites	pixmaps composed to lettering pixmap combinations to form screens
docs	main package documentation
labels	pixmaps of menu items
lettering	pixmaps of descriptive lettering used on screens
src	code files to generate `e6`

3.2.6 Coding notes

With version 2, the `Note` reference type was implemented as a multi line field. A variable `singleLine` was used to opereate as a single line of the `Note` reference type. The contents of the `singleLine` variable is displayed in the `Note` field on the screen, not the `dataStore[26` in which the Note field text is held. The `singleLine` variable is of the same type as all the `dataStore` variables. The length of the `singleLine` array is the same as that of the `Note` variable `dataStore[26]` to allow for a `Note` field not being split into multiple lines.

The `keyNote()` routine was created to handle multi line input. The `keyEntry()` routine for handling single line input was modified resulting from the experience in coding of `keyNote()`.

The array of the `singleLine` variable forms the `Note` field on the screen, one line at a time. The array of the `dataStore[26]` stores the note which is eventually stored in the reference file. These two arrays are coupled together in the `keyNote()` routine.

The partial appearance of the top part of the insertion cursor at the bottom left hand corner of the `Note` input window is deliberate. It indicates no more lines, and thus characters, can be inserted in this field.

The reference type multi-volume reference is not implemented because the fields it should contain are not defined in the `biblatex` documentation.