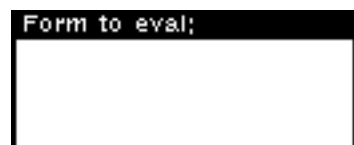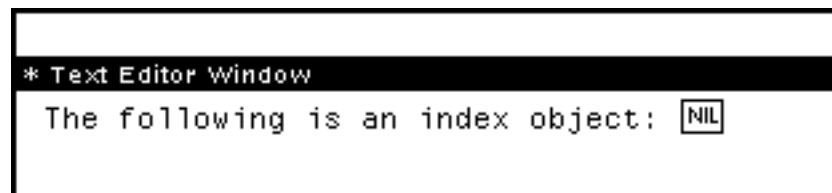## == IMINDEX ==

The file IMINDEX contains the functions used for creating and editing index image objects, and inserting them into a Tedit document. When a Tedit document containing index objects is formatted for printing, the index objects do not appear, but information about the index objects is put into an auxilliary "IMPTR" file. The functions in IMTOOLS can be used to take a set of IMPTR files, and generate an index.

Adding an Index Object to a Tedit Document

The simplest way of adding an index object to a Tedit document is to type ctrl-O while typing at Tedit, which will cause a window to pop up asking you to type a form to eval:

Typing (IM.INDEX.CREATEOBJ) will create an empty index object, indexing the term NIL, inserted in the Tedit document at the caret. Index objects appear in the Tedit window as words with boxes around them:
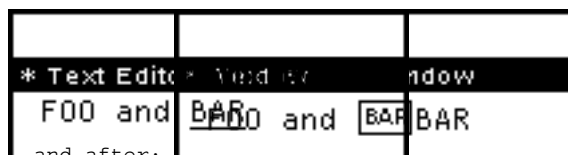

Using the IM Index Menu

An easier way to put many index objects into a Tedit document is to type (IM.INDEX.MENU) at the lisp exec, which will prompt you to position a menu that looks like this:

If the caret is blinking in a Tedit window, selecting [Index Selection as Term] with the left button will create and insert an index object that indexes the selected string. For example:

Before: and after:

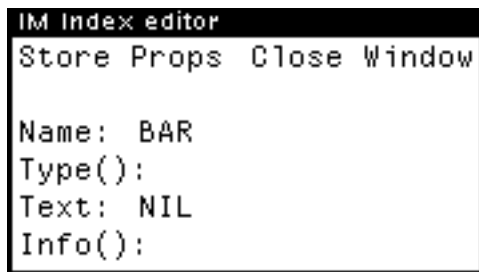Selecting [Index Selection as Term] with the MIDDLE button will insert an index object that indexes the selected string, and then create an index object editing window, that can be used to edit the values of the fields in an index image object.

Editing an Index Object in a Tedit Document

If you select an index object, a menu will appear asking whether you want to edit the contents of the index object:

If you select [Edit Index], you will be prompted to position an index object editing window, which looks like the following:

```
IM Index editor
Store Props  Close Window


Name: BAR
Type():
Text: NIL
Info():
```

This is a freemenu that allows you to edit the values of the fields in an index image object. Selecting [Store Props] stores the values in the editor into the object itself -- if the "Name:" field is changed, the Tedit window will change to show the new index name. Selecting [Close Window] will close the editor window. Note that any changes are lost if you close an index object editing window without storing the new property values.

Index Object Properties

There are more properties in an index object than show in the window initially; the window can be scrolled or reshaped to see and edit the other properties. The fields are interpreted as follows:

Note: The properties listed with "()", such as "Type():", interpret their value as a list of take a list of items, delimited by spaces. The other properties interpret their value as an atom, including spaces.

Name: Value is the name used to sort and merge index entries. This should normally
                    be all-uppercase. The [Index Selection as Term] item in the IM Index menu
                    will automatically uppercase the selection if it is not all-uppercase, and
                    put the real value in the "Text:" field. Examples:
                    Name: FOO
                    Name: BAR

Type():   Value is the type of object being indexed. If NIL, this stands for an
                    English term. Other types are used for indexing lisp functions, variables,
                    etc. Note that upper/lower case is important. Examples:
                    Type(): Function
                    Type(): Editor Command

                    Note: The special types CHAPTER and SUBSEC (all-uppercase) are used to
                    create entries in the table of contents, as described below. These cannot be
                    used as index entry types.

Text: Value is the name actually printed in the index. If NIL, the value of the
                    "Name:" property is used. This is usually used when indexing non-uppercase
                    items. Another place where this is useful is when you want an item
                    containing strange characters to appear in a different place in the index.
                    For example, if Name: = FOO and Text: = *FOO*, the item "*FOO*" will appear
                    in the index amoung the "F" items instead of the "*" items.

Info():   Value is a list of "information" words, that mean something to the indexing
                    programs. There are a few info words likely to be used by IMINDEX users.
                    One is the word *PRIMARY*, which indicates that this is a primary reference.
                    In the index, primary index page numbers are printed first, in a bold font.
                    Another is the word *NOPAGE*, which indicates an index entry that should not
                    be printed with a page number. This can be used to generate entries such as
                    "FOO, see BAR". Example:
                    Info(): *PRIMARY*

SubSec():   Probably not much use for IMINDEX users. Value is a reverse list of nested
                    subsection and chapter numbers. For example,
                    SubSec(): 4 2 99
                    indicates subsection 4 inside subsection 2 inside chapter 99. This is not
                    used with normal index entries, except that chapter numbers, if given, are
                    used when generating the page number in the index. For example, if the index
                    entry specified chapter 99, and it was on page 5, it would appear in the
                    index as 99.5.

                    Another use for this field is if you use IM index objects to generate a table
                    of contents, by creating index objects with type of CHAPTER or SUBSEC, as
                    described below. In this case, the SubSec field is used to specify the
                    subsection numbers.

Page#:   The value of this field is replaced with the page number when the Tedit
                textstream containing the index image object is hardcopied. Not much use to
                change, but it is possible.

SubName:
SubType():
SubText: Name, Type, and Text values for a sub-entry, if any, which appears under the
                main entry in the index.

SubSubName:
SubSubType():
SubSubText: Name, Type, and Text values for a sub-sub-entry, if any.

Adding New Types to the IM Index Menu

If all of the index entries in a document are terms, it is very easy to use the IM Index menu to
index them. However, if there are a lot of Functions, variables, etc., you may need to edit each
index entry to change the type. To make this easier, you can use the [>>Add Type<<] button in
the IM Index menu to add new selections to the IM Index menu. If [>>Add Type<<] is selected, the
system will prompt you to type a type name, and add a new item to the menu. For example, if you
started with the initial menu, and added a new entry for the type "Function", the menu would be
changed to look like the following:

If [Index Selection as Term] is selected, an index object will be created indexing the the
current Tedit selection, with the type of Function.

Saving and Retrieving Tedit Documents with IM Index Objects

Tedit documents containing IM Index objects can be saved using the ordinary Tedit "Put" command,
which will store all of the indexing information in the index objects. Note, however, that it is
necessary to load the IMINDEX package before editing any Tedit documents containing IM Index
objects -- otherwise Tedit will print the message "WARNING: Document contains unknown image
objects", and all index objects will appear as the following:

If this happens, it will not be possible to use these index image objects, or re-save the Tedit
document. Close the Tedit window, load IMINDEX.DCOM, and call Tedit to edit the document again.

Hardcopying a Tedit Document Containing Index Objects

A Tedit document containing index objects can be hardcopied using the normal [Hardcopy] menu
button, or from the Filebrowser. Index image objects will not appear in the final hardcopy.
However, while the document is being formatted, the index objects will put indexing information
in an auxilliary file with the extension "IMPTR". The first time an index object is formatted,
it will try to open an imptr file on the connected directory, printing a message in the prompt
window: "Opening index pointer file: {DSK}<LISPFILES>FOO.IMPTR...done". When the formatting is
completed, another message will appear: "Closing index pointer file:
{DSK}<LISPFILES>FOO.IMPTR;1...done".

The directory used for the imptr file is the currently-connected directory (which can be changed
using the CONN command). The file name used if the file name of the Tedit file, if there is one.
For example, if the Tedit file is XYZ.TEDIT, the imptr file would be XYZ.IMPTR. If the Tedit
text stream doesn't have a name (for instance, if the Tedit window has been brought up but never
saved) the file name NONAME.IMPTR is used. The extension is always IMPTR.

For Hackers Only: If there is a need to use a different filename for the imptr file, it is
possible to specify this programmatically by changing textstream properties of the Tedit
textstream before the hardcopy operation. If the value of the textstream property
IM.INDEX.PTRFILENAME is non-NIL, it is used as the file name for the IMPTR file. If the value of
the textstream property IM.INDEX.PTRFILE is non-NIL, it is the stream used for the IMPTR
information.

Creating Indicies and Tables of Contents From IMPTR Files

The file IMTOOLS.DCOM contain the routines used to create indecies and tables of contents for documents in IM format. These will also produce indicies and tables of contents from IMPTR files created by IM index objects. Loading this file will automatically load IMINDEX.DCOM, if it is not already loaded. Note, however, that you can load IMINDEX (for the purpose of editing Tedit documents with IM index objects) without loading IMTOOLS.

IMTOOLS contains the following useful functions:

(MAKE.IM.INDEX OUTFILE.FLG VOLUME.INFO IMPTR.FILES IMPTR.TYPES)   [Function]

MAKE.IM.INDEX takes a number of IMPTR files, and produces an index, formatted like the one in the IRM. If OUTFILE.FLG is NIL, the output file is just sent to the default printer. If OUTFILE.FLG is T, the outfile textstream is simply returned. If OUTFILE.FLG = anything else, it is taken as a file name of an interpress file which is created <but not printed>.

VOLUME.INFO is a list of lists specifying which chapters are associated with which volumes in a large multi-volume document (like the IRM). For an example of the format, see the variable IM.MANUAL.VOLUMES in IMTOOLS. If NIL, no volume numbers are printed in the index.

IMPTR.FILES is a list of IMPTR files to be used to create this index. If NIL, any index info loaded in by explicitly calling GRAB.IMPTR is used (see IMTEDIT.TEDIT). When doing the IRM, all of the IMPTR files are loaded into global variables using GRAB.IMPTR, but users of IMINDEX probably want to specify the values specifically.

IMPTR.TYPES should be a list of IM index types to be included in the index, specified as lists. For example, if IMPTR.TYPES = ((Function)(Variable)), only function and variable entries would be listed in the index. If IMPTR.TYPES = NIL, all index entries in the IMPTR files are used.

(MAKE.IM.TOC OUTFILE.FLG CHAPTER.NUMBERS IMPTR.FILES)   [Function]

MAKE.IM.TOC takes a number of IMPTR files, and produces a table of contents, formatted like the one in the IRM. All im index entries whose type is CHAPTER or SUBSEC are used as chapter and subsection pointers.

OUTFILE.FLG and IMPTR.FILES are interpreted similar to MAKE.IM.INDEX.

CHAPTER.NUMBERS is either: NIL, meaning to generate TOC of ALL data in the specified imptr files; a single number, meaning to generate a chapter TOC for that chapter; or a list of numbers, meaning to generate a TOC for those chapters.