FetchRefs_GI

COLLABORATORS

	TITLE : FetchRefs_GI	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FetchRefs_GI

1.1 FetchRefs_GI

```
                              FetchRefs 1.2

            A feature packed utility that provides you with a
          comfortable access to your AutoDocs and include files

                           (GenerateIndex manual)

 Table Of Contents

 Introducing GenerateIndex
 Arguments
 Scanning~options
 The~main~window
 The~references~window
 The~options~window
```

1.2 Table Of Contents

1.3 Introducing GenerateIndex

For FetchRefs to be able to look things up just anywhere near fast, it needs an index. The FetchRefs package has a seperate program to generate this index file: GenerateIndex. You need to start GenerateIndex whenever you want to update your index. This is typically whenever you install new or updated docs or includes.

GenerateIndex has both an easy to use GUI for the beginner and enough Shell arguments to make it work automatically (via a Shell script) for the more experienced user.

You can have several index files, but I suggest for your own sake that you keep it at one. As everything is kept together you will know exactly where you have it. If you start making several index files you will probably need to change the ARexx script you use, and therefore you should feel used to FetchRefs before doing this. The suggested and default name for your index file is 'S:FetchRefs.index'.

Requirements

1.4 Requirements

GenerateIndex takes advantage of
- reqtools.library, Copyright 1994 by Nico François
- triton.library, Copyright 1995 by Stefan Zeiger

Thanks to both authors for creating and releasing these libraries!

GenerateIndex requires at least version 38 (release 2.x) of ReqTools and version 5 (release 1.4) of Triton. Furthermore, you need an Amiga with Kickstart 2.0+ and preferably a harddisk and a fair amount of RAM.

GenerateIndex is operateable in script mode (from the Shell) without these libraries but you will need Triton anyway for FetchRefs and you will probably have ReqTools installed.

I have included triton.library and the installation script will take care

of installing/updating it if you do not have a more recent version. ReqTools is not included but available everywhere.

1.5 Arguments

The Shell template for GenerateIndex is:

```
FROM/M,TO,SETTINGS,RECURSIVELY/S,KEEPEMPTY/S,UNRECOGAREDOCS/S,AutoDoc/S,
C/S,C_DEFINE/S,C_STRUCT/S,C_TYPEDEF/S,E/S,E_CONST/S,E_OBJECT/S,E_PROC/S,
ASM/S,ASM_EQU/S,ASM_STRUCTURE/S,ASM_MACRO/S
```

or, for those who prefer BNF:

```
GenerateIndex [[FROM] {wildcard}} [TO <file>] [SETTINGS <file>]
   [RECURSIVELY] [KEEPEMPTY] [UNRECOGAREDOCS]
   [AutoDoc]
   [C] [C_DEFINE] [C_STRUCT] [C_TYPEDEF]
   [E] [E_CONST] [E_OBJECT] [E_PROC]
   [ASM] [ASM_EQU] [ASM_STRUCTURE] [ASM_MACRO]
```

(read the deciding~what~files~to~scan and deciding~what~to~scan~for sections for arguments that do not have links above!)

This is rather confusing unless you know GenerateIndex well. Therefore I suggest that you get used to GenerateIndex by using the GUI before trying to use it without. However, when you understand all arguments you might want to make a Shell script which scans your include and AutoDoc directories with appropiate options and simply execute this script everytime something has changed in these directories. This will save you some time.

If you do *not* specify FROM as a Shell argument, the GUI is opened. Otherwise GenerateIndex will assume script mode and you will have to enter some more arguments, too. When you start from the Shell there are no default values; every option is turned off.

The GUI will always open if you start GenerateIndex from your Workbench. Furthermore, only two arguments are available: TO and SETTINGS.
When shipped, GenerateIndex has a tool type of 'TO=S:FetchRefs.index' set. This matches a FetchRefs tool type of 'FILES=S:FetchRefs.index'. Therefore you will be fine if you edit none of these.
When the SETTINGS tool type is not present (as is the case when shipped), GenerateIndex will read the file 'ENV:FetchRefs_GI.prefs'. This is where the gadgets 'Save' and 'Use' in the~options~window save the settings so this should be fine, too.

Basically, beginners will start from Workbench and have most things set up acceptable. Experts can start from Shell and customize everything.

Below is a short description of each argument.

FROM
TO
SETTINGS

1.6 FROM

If you specify the FROM argument it must be a specification of what files to scan for references (i.e. either include files or AutoDocs). You can specify as many files as you wish and if you specify the argument RECURSIVELY you may also specify a directory (containing e.g. all your include files). Note that you are actually specifying wildcards so e.g. "INCLUDE:*.h" is perfectly valid. This argument is the same as selecting files by using the Scan... gadget of the~main~window of the GUI.

The FROM argument is not accessible via Workbench tool types - instead the GUI is provided. This is because FROM is intended to be used in a script that automatically recreates your index file; this script can then be run when you get new manuals or include files.
This is also why all the options arguments are only available for Shell usage - they are mainly useful in a script, otherwise you would prefer the GUI to set them.

An example script for creating an index file is provided (GenIndex.sh), but this example is very specific to my setup and really IS just an example - you will have to make one for yourself if you want one!

1.7 TO

The TO argument specifies an index file. If this file exists it will be loaded and any newly scanned references will be appended to it. If the file does not exist, the name is just used as the file name to save to.

TO is also valid as a tool type but you probably will not want to change it from the value it has when shipped.

WARNING: It is a very bad idea to specify a non-FetchRefs file as the file format of FetchRefs does not currently include an ID. GenerateIndex will interpret anything specified as a data file and may crash in the attempt!

1.8 SETTINGS

The settings argument lets you specify a previously saved settings file to load upon startup.

If you start GenerateIndex from Workbench but do not specify this tool type, the items in the~options~window will get set to the settings saved in 'ENV:FetchRefs_GI.prefs'. As SETTINGS only changes what settings file you *load* it is not too wise to change it; 'Save' and 'Use' in the~options~window will always save to 'ENV:FetchRefs_GI.prefs'. Effectively, if you use the SETTINGS tool type you cannot use 'Save' and 'Use' but will have to use the menu to specify the save name specifically.

There is no default settings file if you start from Shell. This is because the other arguments only let use *activate* options - therefore you have to start out with no options and activate just the ones you prefer. If you do not want to enter you options everytime you scan, you can save them

once and for all via the GUI and simply use the Shell argument SETTINGS to load them. This just prohibits you from altering any of the options by hand. If this is too hard to understand, start from Workbench.

1.9 Scanning options

GenerateIndex has a lot of scanning options. They allow you to configure it to work exactly as you want. At first sight the many gadgets (and even more the many Shell arguments) may seem a bit frightening. However, the system is quite simple once you learn to operate them.

I will not list every gadget and argument name here. Instead, I will simply explain the system - that should be sufficient to learn you how to operate GenerateIndex.

Before scanning you need to decide three things: what kind of files to scan, what to look for inside these files, and finally what special options you want. You tell GenerateIndex what you want to do either by activating the gadgets in the Options window or by entering the Shell arguments.

The scanners are not very intelligent and therefore you may experience problems if you scan something that is not entirely standard or a bit complicated (like wild C typedef statements or AutoDocs with missing form feeds between entries). Please report any problems you have, otherwise I will probably not improve the scanner. However, I will never "improve" the scanner to accept invalid constructions.

1.10 Deciding what files to scan

The first thing GenerateIndex does to a file is checking its type. This is done by checking if some of the following keywords appear:

```
          AutoDoc: "TABLE OF CONTENTS"

        C include: "#if", "#define"

        E include: "\n(---) OBJECT", "\nCONST", "\nPROC"

Assembler include: "IFND", "EQU", "STRUCTURE", "MACRO", "BITDEF"
```

Furthermore, if unrecognized files are set to be treated as AutoDocs (see UNRECOGAREDOCS), all other files will be identified as AutoDocs.

The supported E include files are the output of ShowModule. This is necessary because the E modules are binary and there is no use in looking

something up in a binary file. I suggest that you build up a directory tree which is a sister to your EMODULES: directory. However, this new tree should contain all the output of ShowModule, one file of output for each module.

Once the file type has been established, the file is either scanned or skipped. If you have activated the file type it will be scanned, otherwise it is skipped. You activate file types by the top gadgets in the options window (above the horizontal bar) or by the Shell arguments 'AUTODOC', 'C', 'E', and 'ASM'. Notice that if a file which is already scanned is scanned again, the first entry is replaced by the new one. This ensures that you do not get any duplicate references even though you scan the same file twice (perhaps because you got a new version).

The normal is to activate 'AUTODOC' as well as whatever language(s) you are using.

1.11 Deciding what to scan for

All the include files that GenerateIndex scans can contain several different types of references. For example, C include files have '#define', 'struct', 'union', and 'typedef' entries. For some reason (e.g. memory consumption considerations) you may want to leave some kind of references out from the index files.

If KEEPEMPTY is turned on, it is *not* the same to deactivate a file type completely and to activate it but leave all types of scanners off! If you activate a file type but do not select any type of references for it, the KEEPEMPTY option will determine whether the files of this type will end up in the index file anyway (but without any references, naturally).

AutoDoc files only contain one type of references and therefore you cannot toggle parts of the AutoDoc scanner on and off.

1.12 Special options

GenerateIndex has a few index options that has nothing to do with the kind of types of files it is scanning. Actually, these options concern the workings right before the file type is determined and right after the scanning has completed.

RECURSIVELY
KEEPEMPTY
UNRECOGAREDOCS

1.13 RECURSIVELY

With this option you can specify if you want the directory scan to be recursive. That is, if GenerateIndex during its directory scan comes to a directory, should it look into it for more files?

The scanner will always include directories you specify yourself. This option only toggles whether any directories *inside* specified directories should be scanned.

1.14 KEEPEMPTY

This option toggles whether files that contains no references should be included in the index anyway. This is useful to activate if you use the FILEREF option of FR_GET (in FetchRefs).

For example, if you have turned scanning for C #define's and typedef's off, <exec/types.h> would contain no references as it contains no structs or unions. But if you search for 'types' you will still want types.h to show up as a file; therefore the file must be included even though it contains no references.

On the other hand; if you do not use the FILEREF option of FR_GET (see FetchRefs_FR.guide), there is no reason to include empty files - they will just make the index file bigger.

1.15 UNRECOGAREDOCS

If GenerateIndex is unable to figure the kind of a file out it can either just forget the file or it can treat it like an AutoDoc.

If you scan AutoDocs that lack the "TABLE OF CONTENTS" line in the beginning (which is wrong) then you should set this option as they will not be scanned otherwise. On the other hand, if you are not scanning AutoDocs you will not want GenerateIndex to try to read other files as AutoDocs as this can be a rather confusing experience for GenerateIndex.

Generally you should only use this option if you experience problems when scanning AutoDocs. The long term solution is to get the author of the AutoDoc to correct it!

1.16 Example of options

A typical choice of options would be:

```
 GenerateIndex RECURSIVELY KEEPEMPTY AUTODOC C C_DEFINE C_STRUCT C_TYPEDEF
```

and to scan your INCLUDE: drawer as well as whatever drawer you have your AutoDocs in.

The complete Shell command line would be (one line, naturally)

```
 GenerateIndex FROM INCLUDE: DOCS:AutoDocs TO S:FetchRefs.index
RECURSIVELY KEEPEMPTY AUTODOC C C_DEFINE C_STRUCT C_TYPEDEF
```

Using the GUI, you would activate 'AutoDocs', 'C includes', '#define', 'struct/union', 'typedef', 'Scan drawers recursively', and 'Keep files without references' in the options window. Then you would select Scan... in the main window and select all files and drawers in the 'INCLUDE:' assign as well as the 'AutoDocs' drawer in the 'DOCS:' assign (this would require you to use Scan... two times).

The above would collect all AutoDoc entries as well as all the information GenerateIndex understands from your C includes. Because of the RECURSIVELY option you do not need to specify every single file.

If you do not understand the system yet, I suggest you try scanning a few small files and see in the listviews of the GUI how much GenerateIndex collected.

1.17 The main window

The main window is the window that is first opened when you start GenerateIndex. This window is covered by a listview gadget containg the names of all the files you have indexed so far.

Below that is a little container showing how many references the currently selected file contains. This is just a quick indication of what references GenerateIndex has scanned in that file. However, if you click on the arrow image next to the container you will open the~references~window which contains a lot more information about the file you have selected here in the main window. The~references~window is also opened if you double click on an entry in the listview.

At the bottom are some gadgets and if you press the right mouse button you will find some menus.

Gadgets
Menus

1.18 Gadgets

Until you have scanned some files, most of the gadgets will be ghosted. This is because they act one the currently selected (or all) files and as you have no files this is not possible.

Scan...
Delete
Options...
Rescan
Rescan all

1.19 Scan...

This gadget will open a file requester. Any file you select here will get scanned and added to the list. Depending on the kind of file it is and how you set the scanning~options, the actual scanning procedure may vary.

You can select several files in one go: hold down a shift key and select all the files you please by clicking on each of them a single time. It is not possible to select files in different levels of the directory hierachy - to do this, you will have to select 'Scan...' several times.

If you scan a file that has already been scanned, the original entry is removed from the list and replaced by the new one.

1.20 Delete

Without any warning, this gadget deletes the currently selected file from the list. At some time you may need this gadget though it is most likely not very often.

1.21 Options...

This gadget opens the~options~window which will let you manipulate the scanning~options.

1.22 Rescan

This gadget will scan the currently selected file again, with the options set at the time you select rescan. This could be used if you get a new version of just one file and do not want to waste time scanning everything again.

A thing to note is that the file could disappear if you rescan it with some options that produce no references and do not have the 'Keep~files~without~references' (KEEPEMPTY) option set.

1.23 Rescan all

Selecting this gadget is like clearing the file list and then selecting Scan... to scan the same files again. This can be used when you have new versions of some files but *not* if you have installed an entirely new package; 'Rescan all' will not notice that new files have turned up in your various directories.

'Rescan all' is also an easy way out if you find out that you have set the options wrongly: Simply correct the settings and 'Rescan all'. This way you save yourself from the trouble of selecting what files to scan once more.

1.24 Menus

The following menu items are attached to the main window of GenerateIndex.

```
Clear
Load~data...
Save~data...
About...
Quit
```

1.25 Clear

This will clear the entire list of indexed files and thus begin a new index file. A warning will be given before flushing the list if you have not yet saved it.

1.26 Load data...

'Load data' will load an index file previously saved with Save~data....
You will be asked for confirmation if you have not saved the current list.

If some files are already indexed, GenerateRefs will ask whether it should clear the list before loading or if it should append the file you specify to the end of the list.

Warning: In the current version, FetchRefs has no ID in its index files. Therefore it can not distinguish index files from anything else but will treat all specified files equally. Loading a non-FetchRefs file is a VERY bad idea!

1.27 Save data...

'Save data' simply saves the entire index to a files which can then later be loaded by FetchRefs or GenerateIndex.

By default, FetchRefs is set to use S:FetchRefs.index as its index file, so by specifying this name you do not have to change any settings for FetchRefs. This is due to the tool types I have defined for you; if you start FetchRefs or GenerateIndex from the Shell, no default name is specified.

1.28 About...

'About' shows the version number and compilation date of the GenerateIndex you are using, as well as my name. What did you expect?

1.29 Quit

'Quit' will abort GenerateIndex. If you have done some changes to the list you will first be asked for confirmation. The exact same action is performed if you click in the close gadget of the main window.

1.30 The references window

This window shows detailed information about what references GenerateIndex has found for whatever file you have selected in the listview of the~main~window. You open it by double clicking on an entry or by clicking on the arrow gadget next to the 'References' container in the main window.

The listview of the top simply contains the names of all the references found.

Below the listview are a few information boxes telling a bit more about the reference you select in the listview. 'File' tells what file this reference comes from and this is naturally the same name as whatever you have selected in the~main~window. 'Offset' and 'Size' specify what part of this file (counted in bytes) the reference covers. For some kinds of references this will be the entire file. 'Offset' will then be zero and 'Size' will be -1. FetchRefs interprets a size of -1 as "the entire file" and figures out by itself how big the files actually is.

'Line' is counted in lines from the start of the reference (Offset) and is the number of lines to jump in the reference to find the start of the "important" stuff.
For example, when indexing C include files, all comments from the previous struct and up till the current is included in the reference. This is because the comments before a struct definiton often contain useful notes. However, the most important information is the struct itself and therefore 'Line' will be equal to the number of lines of comments before the struct. FR_GET (see the FetchRefs_FR.guide) will return this number upon a successful reference look-up so the ARexx script can figure out where to position the cursor.

The gadget 'Delete' at the bottom will delete the selected reference. This is useful if you have several references of the same name but only really need one of them - this way you can delete the others and keep FetchRefs from popping up its window, asking which one you want. The downside of this is that you will have to delete them once more everytime you rescan.

1.31 The options window

This window contains many gadgets to configure the scan phase of GenerateIndex to act exactly like you want it to. You should read the section scanning~options to learn how to use these gadgets.

Furthermore there is a little menu to enable you to load and save your settings. This is only needed if you need more than one set of settings.

Otherwise you will use the 'Save' and/or 'Use' gadgets which will save the settings to the files "ENVARC:FetchRefs_GI.prefs" ('Save') and "ENV:FetchRefs_GI.prefs" ('Save' and 'Use'). You can get back the last set of options that you saved with the 'Save' gadget by selecting the menu item 'Last saved'.

The 'Cancel' gadget and close icon of the window will both leave the options window and set the options back to what they were when the window was first opened.

This should all be close to the standard Preferences behaviour and hard to misunderstand ;-).