FetchRefs_FR

COLLABORATORS

	TITLE : FetchRefs_FR	

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FetchRefs_FR

1.1 FetchRefs_FR

```
             FetchRefs 1.3

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

          (FetchRefs manual)

 Table Of Contents

 Introducing FetchRefs
 Arguments
 ARexx~commands
 The~<Select~reference>~window
 ARexx~scripts
```

1.2 Table Of Contents

1.3 Introducing FetchRefs

FetchRefs is the heart of the package and you will probably want to have it running always. Anything FetchRefs does is based on ARexx commands, be it load a new file, flush its buffer, quit, or do its best to find documentation on whatever (part of a) keyword the caller supplies.

Before FetchRefs is any good, it needs an index file. This is generated by GenerateIndex. Please be sure to read the file 'FetchRefs_GI.guide' to learn everything about generating index files! Also be sure that you *have* an index file before you try to use FetchRefs.

If you decide to skip parts of this guide, please make sure that you read at least the FR_GET and the the~<Select~reference>~window sections.
Thanks a lot.

Requirements

1.4 Requirements

FetchRefs requires
 · triton.library, Copyright 1995 by Stefan Zeiger

Thanks to the author for creating and releasing this library!

You will need at least version 5 (release 1.4) of Triton and naturally an Amiga with at least Kickstart 2.0. Finally you will need to have ARexx running.

1.5 Arguments

The syntax of FetchRefs is (in standard AmigaDos notation):

```
  FILES/M,PORTNAME
```

or, if you prefer this BNF-like approach,

```
  FetchRefs [[FILES] {wildcard}] [PORTNAME <name>]
```

As you can see(?), you can specify as many index FILES as you please.
If you want to use tool types (i.e. start from Workbench, possibly via the WBStartup drawer), you simply enter each argument as a tool type. To enter several index FILES you simply make several tool type entries, all starting with 'FILES='.

In the following sections the arguments are explained.

FILES
PORTNAME

1.6 FILES

The FILES argument specifies what index files FetchRefs should initially load. These files are generated by GenerateIndex and are unreadable for anything but FetchRefs and GenerateIndex. You may enter as many index files as you please, though most people will do fine with just one.

One issue to remember is that what you specify via the FILES argument are not really file names but rather wildcard specifications. This means that you can use all the standard things like # ? [] and so on to specify the FILES. If you enter a wildcard that matches no files, a warning is written to the Shell window or in a requester if you start from Workbench. This is to catch the case where you make a typing mistake - otherwise it can be very hard to figure out why FetchRefs does not load any index FILES. Except from that warning, FetchRefs will just consider it a wildcard with no match and do nothing more about it.

It is not required that you enter any FILES. For example, people with very limited memory might want to have no files loaded and then use the ARexx function FR_NEW to load the index file right before they use FR_GET to fetch the reference. After fetching they will then use FR_CLEAR to flush the index file again.

1.7 PORTNAME

The PORTNAME argument specifies what the name of FetchRefs's ARexx port should be. The default is 'FETCHREFS'.

If the specified (or default if nothing is specified) port already exists, FetchRefs will send it a break signal and exit. Therefore you can quit an already running FetchRefs by simply starting another copy. The new instance will then make sure that they both quit. You can start FetchRefs for a third time after this and it will then stay in memory.

If you really want several instances of FetchRefs running at the same time you need to give each copy a different PORTNAME.

1.8 ARexx commands

FetchRefs is operated through a few ARexx commands. For the basic usage of FetchRefs you do not need to know particulary much about any of these. Nevertheless, you will probably want to know how and why FetchRefs does what and when if you want to customize it a bit.

In any case you should read the FR_GET section as it covers a big part of FetchRefs.

The file 'FetchRefs.ADoc' (without icon and not installed by Installer script) contains the documentation on all the ARexx commands in standard AutoDoc format. Thus you can scan it with GenerateIndex and (with FetchRefs) quickly look up the manual pages if you should need them while editing an ARexx script.

```
 FR_ADD
 FR_CLEAR
 FR_GET
 FR_NEW
 FR_QUIT
```

1.9 FR_ADD

```
FetchRefs/FR_ADD              FetchRefs/FR_ADD

   NAME
  FR_ADD -- load additional index files

   SYNOPSIS
  FR_ADD FILES/M

  FR_ADD [wildcard [...]]

   FUNCTION
  FR_ADD will load extra  index files and add  them to the  internal
  list. The index files already in memory are not removed.

   INPUTS
  FILES/M - wildcard specification(s) for the index files to load.

   RESULTS
  None.

   BUGS
  None known.

   SEE ALSO
  FR_CLEAR, FR_NEW
```

1.10 FR_CLEAR

```
FetchRefs/FR_CLEAR              FetchRefs/FR_CLEAR

   NAME
  FR_CLEAR -- remove any index files from memory

   SYNOPSIS
  FR_CLEAR

   FUNCTION
  Frees all memory allocated  to store loaded  index files. Most  of
  the memory FetchRefs uses is the index so this will put  FetchRefs
  into a low-memory sleep  mode. By later  calling FR_ADD or  FR_NEW
  the original state can be restored.
```

```
   INPUTS
  None.

   RESULTS
  None.

   BUGS
  None known.

   SEE ALSO
  FR_ADD, FR_NEW
```

1.11 FR_GET

```
FetchRefs/FR_GET               FetchRefs/FR_GET

   NAME
  FR_GET -- get a reference into a file or the clipboard

   SYNOPSIS
  FR_GET FIND/A,TO/A,PUBSCREEN,FILEREF/S,CASE/S

  FR_GET <keyword> <filename> [public screen name] [FILEREF] [CASE]

   FUNCTION
  Searches the index list for a  name matching the FIND keyword  and
  writes it to the file specified by TO.

  The FIND argument  is a wildcard.  Thus you can  search on  things
  like 'Open#?'  and get  a  long list  of functions  starting  with
  'Open'. Many more wildcards exist; all the standard AmigaDOS  ones
  are accepted.  However,  though  FR_GET  supports  wildcards,  the
  provided ARexx scripts do not!

  If you need wildcarding capabilities you  can put the cursor on  a
  space (see below)  before executing  the ARexx  script. This  will
  open the~<Select~reference>~window  where all  wildcards can  be
  used in the pattern string gadget.

  When no matches are found, an error is returned. If excatly one is
  found, the reference is written  to the filename specified by  the
  TO argument.

  If the  supplied  keyword/wildcard  turns  out  to  match  several
  references, a window is opened. Read more about this window in the
  section named the~<Select~reference>~window, please. The  window
  will also open if the empty  string '' matches the wildcard.  This
  means that you can  position the cursor on  a space character  and
  invoke FetchRefs. Then the  window will open and  you can enter  a
  search pattern. Useful if you are going to look something up  that
  is not in your current source view.

  The screen on which the above mentioned window is to be opened  is
  specifed by using the PUBSCREEN argument. You specify the name  of
```

a public screen which may not be in private mode (they rarely are). If the specified screen is not available (non-existent or non-public) or if you do not specify PUBSCREEN at all then FetchRefs will open the window on the currently active screen. Should this not be public, the default public screen (usually Workbench) is used.
No matter where the window opens that screen will be brought to front (if it is not already there). When you have finished the selection and the window closes, the screen is again put behind the other screens (but only if it was brought to front in the first place).

If the FILEREF argument is given, each of the files in the index file will be considered a reference themselves. The name of the reference is the filename without any leading path or suffixes. For example, the file 'DINCLUDE:Amiga30/exec/types.h' would be considered a match if you search for the reference 'types'. The reason for this truncation is mainly due to the way FetchRefs works otherwise; types.h must be truncated at the dot if 'types' was a C structure and not a file name - and FetchRefs really has no way of knowing what it actually is, until the match is already found; so, the most sensible idea seemed to trucate everything at the first non alpha-numeric character.

Depending on whether you set the CASE flag or not, the comparison of the reference names is either case sensitive (CASE specified) or not. If you are very good at memorizing capitalization you may want to turn it on - personally I prefer to keep the search case insensitive.

To write the reference to the clipboard instead of a file, a filename of 'CLIPnn' can be specified. nn is the number of the clipboard unit you wish to use. The 'CLIP' word must be in uppercase, otherwise the name is considered an usual file name.

 INPUTS
FIND/A - name of reference to search for. Wildcards accepted.
TO/A - file name to put the result into. 'CLIPnn' specifies
the clipboard unit nn.
PUBSCREEN - public screen to open the "select reference" window
on. Default is the currently active screen (if public,
otherwise the default public screen).
FILEREF - let a reference search on the base name of a file
match with the entire file.
CASE - activate case sensitive search.

 RESULTS
Two results are returned.

rc will be
0 if the reference was successfully written
5 if the~<Select~reference>~window was cancelled/closed
10 if no match was found for the specified search pattern
20 if an error (no memory, etc.) happend during the fetch

rc2 contains additional information; if rc is 0 then it will contain a number specifying what line the editor should move the

```
  cursor to after having loaded  the generated file. This line  will
  contain the core of  the requested reference. If  rc is 5, 10,  or
  20, rc2 will be a string  describing what went wrong. This can  be
  passed on to the user through the editor (requester, title line).

   BUGS
  None known.

   SEE ALSO
```

1.12 FR_NEW

```
FetchRefs/FR_NEW               FetchRefs/FR_NEW

   NAME
  FR_NEW -- clear internal index list and load a new

   SYNOPSIS
  FR_NEW FILES/M

  FR_NEW [wildcard ...]

   FUNCTION
  This is a combination  of FR_CLEAR and FR_ADD  and results in  the
  internal list being  set to  nothing but what's  specified by  the
  FILES arguments.

   INPUTS
  FILES/M - wildcard specification(s) describing what files to load
      instead of the current list.

   RESULTS
  None.

   BUGS
  None known.

   SEE ALSO
  FR_ADD, FR_CLEAR
```

1.13 FR_QUIT

```
FetchRefs/FR_QUIT              FetchRefs/FR_QUIT

   NAME
  FR_QUIT -- force FetchRefs to quit

   SYNOPSIS
  FR_QUIT

   FUNCTION
```

```
  Will send a ^C signal to the FetchRefs process that owns the ARexx
  port. This  will force  FetchRefs to  free all  allocated  memory,
  close down the ARexx port, and exit.

  A similar  effect can  be achived  by using  the C:Break  program,
  running FetchRefs again, or by sending a ^C by any other means.

   INPUTS
 None.

   RESULTS
 None.

   BUGS
 None known.

   SEE ALSO
```

1.14 The <Select reference> window

FetchRefs hates being wrong. It will much rather be stupid than wrong. Therefore it will ask you whenever it gets in doubt. You can bring FetchRefs in doubt by giving it ambigious orders. If you ask it to fetch a reference but leave several posibilities open, FetchRefs will ask you to specify what you really want.

There are three ways to get FetchRefs asking: using a wildcard that matches several references (like 'Open#?'), fetching something like 'DateStamp' or 'OpenDevice' which is documented in several places, and finally asking to fetch a reference that matches '' (i.e. place the cursor on a space and invoke the GoFetchRefs ARexx script).

FetchRefs asks by opening a window with a listview. This listview contains all the references that match whatever you asked for. Below this listview is a string gadget which contains your search wildcard. Both of these will naturally be empty if you placed the cursor on a space.

All references in the listview are picked with the options that are given to the FR_GET call which got the window opened in the first place. In other words, you cannot change the FILEREF and CASE options while the window is open.

If you see a reference you like in the listview you can double click on it and the window will disappear and your editor will soon have the reference ready. You can also navigate in the listview by the arrow keys and the numeric keyboard and select to load the selected reference by using Enter. The gadget 'Fetch reference' also means the same as double clicking on an item.

If you do not want any of the references in the listview you can press the 'Cancel' or the close window gadget or the Esc key and the window will disappear and you will be back in your editor right where you left it.

You can also enter a new search pattern in the string gadget and then the contents of the listview will change to the references that match the new

pattern. This is quite powerful if you search for a function but are not quite sure what the name is; you can simply try until you see it in the listview and then fetch it. Note that while the string gadget is active, pressing return will *not* fetch the reference selected in the listview. Instead, it will confirm the contents of the string gadget and switch the contents of the listview.

As a final possibility, you can press the 'List file' gadget whenever you have selected an entry in the listview. This will change the contents of the listview to all the references in the file which the selected reference also comes from. This is useful when you are looking for functions in "family" with a certain function but you do not know their names. Simply get the function name you know into the listview by writing it in the string gadget and then press 'List file' - suddenly you have all functions in the same library as the one function you know and chances are that you will now find the name of the related function you are looking for.
You can also use this gadget to list all functions in any AutoDoc, if you use the FILEREF option of FR_GET. Simply enter e.g. 'intuition' in the string gadget, press Enter, select 'intuition.doc' in the listview, and press 'List file'. Now the listview will contain a list of all the functions in the Intuition AutoDoc.

1.15 ARexx scripts

Included in the FetchRefs distribution are some pre-made ARexx scripts for popular editors (the list is shown by the Installer script).

I consider it your problem to figure out how you execute the script from your particular editor. The basic idea, however, is to assign the command "execute arexx script 'GoFetchRefs'" to a key and then simply press that key when your cursor is on top of the word you want a reference for.

If you want to change the options of FetchRefs (as described in FR_GET) you must locate the line starting with 'FR_GET' and change it according to your needs. This line is present in all the scripts.

If you optimize a script or invent an entirely new script for an editor that is currently not yet supported, I am interested in getting a copy so I can distribute it along with further versions of FetchRefs that might be. Please note, however, that various extensions to make the scripts meet your needs are not something that I am generally interested in. The scripts are *meant* to be simple.

The~script~for~the~Shell

1.16 The script for the Shell

Apart from all the scripts for editors, a generic one is also supplied. This script does not need any editor and just prints the reference to a Shell window. You execute it from a Shell prompt like this:

```
  Shell> rx GoFetchRefs OpenScreen
```

Of course you change 'OpenScreen' to whatever you want to search for. Wildcards are accepted, so you could go 'rx GoFetchRefs (Open|Close|%)Scr#en' - if you should happen to know these wild wildcards ;-).

You can also make a little script that fetches the reference and brings it into your favourite text viewer - but I will leave that as an exercise to you.