in

Benjamin Rogers

COLLABORATORS			
	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Rogers	March 1, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 IconAdder

Icon Adder

v0.95ß

1.2 Introduction

This program is like other 'icon adder' programs you may, or may not, have seen before. It uses the WhatIs.Library to find the file type, and then adds an icon according to it's type.

The features include:

- Option to replace existing icons.
- Can choose which drawer icon to use via the ReqTools file requester.
- Add icons belonging only to files or drawers.
- 'Float' the new icons, letting Workbench find the best place for it.
- Add icons only to a particular type of file.
- Suppress any messages from the program.
- Can be run from WB or the CLI. From WB it will open a AppIcon, where files and drawers can be dropped on, to change their icon. From the CLI, it will process one file.
- Uses the Def_#? icons in the ENV:Sys drawer, enabling you to customise your icons to your preference.
- Select individual drawer icons to use (for those of you that like to have different MWB drawer icons like me ;-)

Requirements:

- ReqTools.Library ©1991-1994 Nico Françis (Included)
- WB 2.x+
- WhatIs.Library © S.R. & P.C. (Included)

To Do:

- Add a GUI for the preferences.
- Choose individual executable icons.
- Copy tool types and default tools from original icon to new icon.
- Choose AppIcon icon (instead of using program icon).

1.3 Copyright

This program has been tested, and so it should not cause any problems. It has been tested on a A1200 (6Mb RAM, 68030, 240Mb HD) and a A4000 (10Mb

RAM, 68040, 250+120 Mb HD, CD-ROM). But, beware, you are using this
program at your own risk. This program comes with absolutely no warranty.
I cannot be held responsible for any damage to you and/or your computer,
or anything else, either directly or indirectly.

1.4 Distribution

The archive should contain the following:

```
     IconAdder (dir)
          Icons (dir)
            Def_AmigaGuide.info             Def_ASCII.info
            Def_Drawer.info                 Def_PPacker.info
            Def_Tool.info
          Libs (dir)
            ReqTools.Library                WhatIs.Library
          S (dir)
            FileTypes                       FileTypes.info
            FileTypes_IconAdder             FileTypes_IconAdder.info
       IconAdder                       IconAdder.Guide
       IconAdder.Guide.info            IconAdder.info
       Icons.info                      Install
       Install.info                    Libs.info
       S.info
  IconAdder.info
```

NOTE: Do not distribute this program without any of these files.

Also, please do not charge any more than necessary for the
distribution of this package, i.e. only to cover the costs of a disk and
postage. This program can also be placed on Public Domain CD's such as
Fred Fish's and Aminet's. Please contact me if you want to put this
archive on a magazine coverdisk.

1.5 Usage

```
               This program can be used from either the
               Workbench
                or the
               CLI/Shell
               .
```

   General Notes:

When the AppIcon is used, double clicking on it will bring up a requester.
On this requester there are three buttons: Quit, Preferences and Cancel.
Selecting Quit will quit the program, selecting Preferences will bring up
the Preferences window, and selecting Cancel will let the program
continue.

   Workbench Usage:

There are seven tool types that are recognised, as follows:

CHOOSEDIR=TRUE|FALSE

This option allows you to choose an icon, via a file requester, for each drawer icon you drop onto the AppIcon. Set it equal to TRUE for this operation, else set it to FALSE to have every drawer get the default system drawer image. Default: FALSE.

FLOAT=TRUE|FALSE

Set this option to TRUE if you want the new icons to be free floating, i.e. stop the icons from having fixed positions. Set it to FALSE for the new icons to have the same position as the Def_xxxxxx icon (in the ENV:Sys/ drawer). Default: FALSE.

NODIRS=TRUE|FALSE

If you only want drawers to be processed, set this to TRUE, else set it to FALSE. NB: The NODIRS and NOFILES options cannot both be set to TRUE, at the same time. Default: FALSE.

NOFILES=TRUE|FALSE

If you only want files to be processed, set this to TRUE, else set it to FALSE. NB: The NODIRS and NOFILES options cannot both be set to TRUE, at the same time. Default: FALSE.

ONLYTYPE=xxxxxxxxx

If you only want one type of file to be processed, set this option equal to it’s type. See also: WhatIs.Library. Default: None.

OVERWRITE=TRUE|FALSE

If you want old icons to be replaced set this to TRUE, else set it to FALSE. Default: FALSE

QUIET=TRUE|FALSE

If you don’t want any messages appearing from the program (apart the Quit requester and the Preferences window) set this to TRUE, else set it to FALSE. Default: FALSE.

DRAWERPATH=xxxxxxxxx

This defines the path to the Drawer Icons, used when the CHOOSEDIR option is used. Default: None.

CLI/Shell Usage:

The template, when running from the CLI is:

```
IconAdder <FileName> [ONLY <FileType>] [DRAWERPATH <Path>] [NOFILES]
          [NODIRS] [FLOAT] [OVERWRITE] [APPICON] [CHOOSEDIR] [QUIET]
```

If you type,

```
   IconAdder ?

a template will appear. Additional help will appear if you get something
wrong on the command line.

The options NOFILES, NODIRS, FLOAT, OVERWRITE, CHOOSEDIR and QUIET are
switches, they act like their ToolType counterparts, apart from that if
they are omitted from the command line they are counted as FALSE.

The ONLY option works the same as the ToolType version, i.e. replace
<FileType> with the file type to be processed. See also:
                WhatIs.Library
                .

Also, the DRAWERPATH works the same as the ToolType version, i.e. replace
<Path> with the directory path for the drawer icons. The Path can be
relative to the directory that IconAdder is run in.

There are two extra options here: <FileName> and APPICON. If the APPICON
option is used, it will cause the program to place an AppIcon on
Workbench, all the options will be used from the command line, and NOT
from the program icon's ToolTypes. The <FileName> option is used to state
which file should be processed. NB: No wildcards or patterns can be used,
only one file at a time. Needless to say, either <FileName> or APPICON
must be specified, and if both options are specified, <FileName> will be
ignored.
```

1.6 History

```
   v0.95ß   17:46:06, 2nd February 1995

First public release.
```

1.7 Author

```
If you want to contact me, up until half way through June 1995, I can be
contacted at:

      Snail Mail:       Benjamin Rogers,
                        Beaumont Hall,
                        Stoughton Drive South,
                        Leicester,
                        LE2 2NA
                        ENGLAND

      E-Mail:           bdr1@le.ac.uk

I can also be contacted at:

      E-Mail:           ben@tibby.demon.co.uk
```

If you want a reply, you are more likely to obtain one if you E-Mail me.
Please send any suggestions about the program and any bug reports
(hopefully there shouldn't be any).

1.8 WhatIs.Library

This is best described by it's own documentation:

WhatIs.library

What is whatis.library ? It is a shared amiga library which alow programmers to easyly recognize type of files (ilbm, 8svx, maxiplan, exe, PP, etc....) BUT the final user (non-programer user) can define new types so all programs which use whatis.library can recognize this new type. At this moment only BrowserII, For and AddIcon use whatis.library but I planned to rewrite Icon (a program which make WorkBench 2.0 recognize files which have no icon), and I encourage all programmers to use this library so users can define new types only one time and every program can use it. I actualy work on version which support the datatype of KickStart 3.0 so user of whatis.library can use both the customisable whatis type and the standard datatype type.

For example, you love graphics, you digitize lots of image with an official commercial program which produce file ".img", and you want BrowserII can reconised it, well, you define this new type in S:FileTypes and whatis.library know what it is, so you BrowserII can do automatic thing on your file ".img", but all program which use whatis.library can reconised the ".img"-type you just defined and can do what they are supposed to do with them.

List of type that whatis.library reconize without any S:FileType file:
DOS_DEVICE VOLUME ASSIGN DIR EXECUTABLE EXECPP40 EXECPP30 EXECPP SCRIPT TEXT OBJECT LIB IFF ILBM ILBM24 ANIM 8SVX SMUS FTXT PREFS TERM ICON IMPDATA PPDATA ZOO LHARC MEDMOD

2)
Well, user. All you need is to know how to define a new type. A sample/starter FileTypes file is provided. First, have a look at it and then read what follows, I think there is not much to say.

There are 2 methods to scan a file: DEEP and LIGHT.
The Light one is only based on the file name and eventualy protection bits. It is fast but unsafe. if you rename an executable as "File.c" and

```
you  ask  for  a  light  scan  (BrowserII  with  "find  type  by  name"),
WhatIs.library think it is a C language source file.
The  DEEP  one  (currently  only one DEEP mode) is more powerfull, but the
slowest  because  each  file  must be open and the first few bytes scanned
(currently 484 bytes), so this slow down directory scans.


FileTypes syntax:

While not necessary, we recommand use of quotes (") delimiters for strings
to avoid mistakes. The # character can be found in strings while it is the
comment starting char.
The "#" char marks the begining of comment until end of ligne.

# example definition:
TYPE "Src Ada"  # you define a new type, it's IDString (curently 9
                # char max) is "Src Ada", this is the string that
                # WhatIs.library will return and you can see in
                # BrowserII  when you ask the "Show file type", it
                # is also the way you identify this FileType.

    SUBTYPE "Text"  # OPTIONNAL: First, the file MUST be a "Text" type,
                    # this means that if the file is not of this type
                    # it cannot be a "Src Ada"

    INSERTAFTER "Script"    # OPTIONNAL: You want the type "Src Ada" to be
                            # put after the "Script" type in list. The
                            # type list is not alphabetically-sorted.
                            # This determine the order in which you see
                            # files when you choose "Sort by file type"
                            # in BrowserII

    ICONNAME "def_Src Ada"  # OPTIONNAL: this the name of the default
                            # icon file name. These files should be in
                            # the "ENV:Sys/" directory, where WB 2.0 put
                            # its default icons. This will be used by
                            # AddIcon (In BrowserII and given cli command
                            # This string is returned by GetIconName()

    # now come the decription of the file, if ANY condition below is
    # not satisfied, the WhatIs.library think it is not this filetype.
    # Exepte for OPNAMEPATTERN which is used for light WhatIs() (only
    # based on the file name)

    NAMEPATTERN "#?.ada"    # OPTIONNAL: if given, the filename must match
                            # this pattern.
                            # it is mutualy exclusive with OPTNAMEPATTERN

    OPTNAMEPATTERN "#?.ada" # OPTIONNAL: same as NAMEPATTERN but it is
                            # a DEEP scan may override it.
                            # it is mutualy exclusive with NAMEPATTERN

    # NAMEPATTERN vs OPTNAMEPATTERN
    # Imagine you are used to name all your image files with .ilbm
    # extension. This way, a LIGHT scan will identify your ilbm files
    # if your specify NAMEPATTERN "#?.ilbm". But ILBM files can also be
    # internally recognized (using DEEP scan). If you specify
```

```
    # NAMEPATTERN "#?.ilbm", all ILBM files not ending with .ilbm will
    # not be recognized by whatis.library. But if you specify
    # OPTNAMEPATTERN "#?.ilbm", the DEEP scan will override the (OPT)
    # name pattern, and all ILBM files will be recognized.

    # Now come the DEEP description. It is the heart of recognition
    # process.  You can specify numbers in decimal (begining with a
    # digit), in hex (begining with $), in binary (begining with "%").
    # Strings begin with a letter or with a quote '"'
    # The search is done within the first (currently 484) few bytes of
    # the file.
    # All these conditions are optional, and are considered as TRUE
    # by LIGHT scan.

    COMPAREBYTE 12 $ABADCAFE    # Test if the file contains the bytes
                                # $AB $AD $CA $FE at offset 12

    COMPAREBYTE $23 "Hello"     # Test if the file contains the string
                                # "Hello" (i.e the bytes $48 $65 $6c $6f)
                                # at offset $23 (decimal 35)

    # in version 2 of WhatIs.library (only under KS2.x) you have an
    # optionnal CASE modifier, this means "A" is different of "a".

    SEARCHBYTE "Good"   # Search for "Good" in the first bytes of file.

    SEARCHBYTE $DEADBEEF    # Search for bytes $DE $AD $BE $EF

    SEARCHPATTERN [CASE] "ST-??:"   # Search for "ST-??:" pattern in file.

    MATCHPATTERN [CASE] 12 "ST-??:" # Search for "ST-??:" pattern in file
                                    # at offset 12.

ENDTYPE     # this marks the end of this FileType definition.
```

AskReparse is a small executable which ask whatis.library to reparse the S:FileTypes file. The file will be effectively parsed only if whatis.library is not used except by AskReparse at call time. Else, the parse is defered until whatis.library has no user.

3)
Look in the WhatIsBase.h, you will find all you want.

How works WhatIs() ?
WhatIs is currently based on 2 methods: light or deep. The light one is only based on the information you pass to it. In deep mode WhatIs() open the file and scans the first few bytes (currently 488: the size of an OldFileSystem data-block), so after loading these bytes in memory, WhatIs() examine them to discover what type it is. WhatIs() also examine the FileInfoBlock.
WhatIs() return a PRIVATE ULONG. You should not make any assumption about how it is coded, because it may and WILL change in future. You keep this ULONG and give it to the different functions of whatis.library. All FileTypes must be first referenced by their IDString "ILBM", "Text", "Exe, etc..., or returned by WhatIs().

For example you want to check if the file "Amiga" is an ILBM picture, you should write:

```
    ULONG Type, ILBMType;

    Type = WhatIsTags("Amiga", WI_Deep, DEEPTYPE, TAG_DONE);
    ILBMType = GedIdType("ILBM");
    if (CmpFileType( Type, ILBMType) == 0)
    {
        /* Yes it is ILBM ! */
        your code here
    }
    else
    {
        /* Not an ILBM */
        your code here
    }
```

Currently supported tags by WhatIs():

```
    WI_FIB      /* TagItem.ti_data = struct FileInfoBlock *FIB, default = NULL */
    WI_Deep     /* TagItem.ti_data = LIGHTTYPE or DEEPTYPE. default = LIGHTTYPE */
    WI_Buffer   /* TagItem.ti_data = Buffer ptr WARNING: your buffer MUST be NUL  ↩
       terminated */
    WI_BufLen   /* TagItem.ti_data = Buffer Len */
    WI_DLX      /* TagItem.ti_data = DLX_numble, found in ArpBase.h */
    /* Version 2.1 or higher */
    WI_DLT      /* TagItem.ti_data = DLT_numble, found in DOS 2.0 */
```

Version history:

```
    1.0:    Version for KickStart 1.3.  Only This Version support 1.3.
            All others need KS2.0
    1.1:    Fixed a little bug.

    ALL next versions NEED KS2.0

    2.0:    First Version of whatis.library.
            (the 1.0 was a limited version made for 1.3)

    3.0:    Fixed a little bug.
            GetParentFileType() Added
            IsSubTypeOf() Added
            DLT support (KS2.0 version of the ARP DLX)

    3.4:    25/12/92
            Fixed a Little Bug: the UNKNOWNFILETYPE was not returned by NextType()

    3.5:    4/1/93
            Fixed a BIG bug born when fixed the preceding bug:
            Forgot subtype of root in FirstType()/NextType().
```

1.9 Installation

The program is installed using the Commodore© Installer utility. This will install the program and this AmigaGuide© file to where you want. It will also ask you which icons you would like to install, out of the collection in the distribution, to your ENV:Sys and ENVARC:Sys drawers, it will also ask which WhatIs.Library FileTypes file you want, either the original or my version (my version having more file types contained). The same installer script can be used to change which FileTypes file is to be used.

NB: Any existing FileTypes file can be backed up during installation.

1.10 ReqTools.Library

```
                    **********************************

                        reqtools.library

                        The requester toolkit.

                        Release 2.2c

                        (c) 1991-1994 Nico François

                    **********************************
```

Whether your program is freely distributable or commercial, you must state in your documentation that your program uses reqtools.library and that ReqTools is Copyright (c) Nico François.

1.11 Index

Workbench Usage