IconManager.doc

COLLABORATORS			
	TITLE : IconManager.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 1, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

IconManager.doc

1.1 IconManager.doc

```
                    IconManager V37.1

      © Copyright 1994 Alex Taylor
           All Rights Reserved


         WhatIs.library

    © Copyright 1990, 1992 S.R & P.C.
           All Rights Reserved


         Contents
         ========
```

1.2 Copyright

DISCLAIMER

You use IconManager at your own risk. No warranties are made or implied. I do
not know of any bugs in IconManager; however, bugs may exist, so watch it!
I cannot and will not be held responsible for any damage or loss of data/life/
family pets caused by the use or misuse of IconManager, so don't bother sueing.

To summarize : no selling, filching, modifying or complaining (unless it's a
bug report).

If you do not agree with any of the above, then you must delete this package
now. THIS MEANS YOU!

1.3 System Requirements

Any Amiga running Kickstart V37+ (not supplied)
WhatIs.library (supplied)

1.4 User Requirements

IconManager is Shareware. If you use it, you are requested to pay the Shareware fee.

Fees are as follows:

£5.00 sterling

OR $US10.00

OR DM15,-

If you live in Great Britain, you can pay by cheque, postal order or cash. For the rest of the world, unless you know an alternative cheap method, please send CASH ONLY! Cash is acceptable in any of the above currencies.

Alternatively, if you write Shareware software yourself, send me a registered copy of your latest program. (Standard density disks only, please.)

See
Contacting The Author
for the addresses.

1.5 Introduction

Since the release of Workbench2, users have had the option to view all files on the Workbench, whether they had corresponding .info files or not. Workbench would create a "pseudo-icon" for those with no icon. However, Workbench only differentiates between six objects : Disk, Drawer, Tool, Project, Trashcan and Kick (non-DOS disk). Additionally, Workbench considers a file to be a Tool if its "executable" bit is set, and a "Project" otherwise, regardless of whether the file is actually a program or not. This is a bit pathetic, not to mention boring, staring at rows and rows of identical icons. Wouldn't it be nice if Workbench could identify the files and display suitable icons for them?

Enter IconManager, stage left :-)

IconManager fools the Workbench into thinking that icon-less files have real icons, and then supplies an icon for each file, based on its type. Simple. Well, not really, otherwise it wouldn't have taken me so long to do... The idea (according to a text file I found whilst clearing my hard drive) was generated last November!

Identifying files is done by use of the whatis.library. It recognises 30 filetypes internally, and more can be added through its configuration file.

See
WhatIs.library

for details. If the file can be identified, and an appropriate icon exists for it, then the icon is passed to Workbench, where it is treated like a real icon. If the file cannot be identified, or if no icon exists, then Workbench uses the standard defaults instead.
Currently, IconManager only fakes icons for files. Drawers and Disks are handled in the normal way.

This is not the first program I have seen that does this trick, but it is the only one that I've seen that actually works...

In addition to this, IconManager provides the ability to write these faked icons out to disk, thus giving the file a real icon. If the file already had a real icon, it will be overwritten, but any of its attributes (position, default tool, tooltypes, etc.) may be preserved.

Finally, if you set the default tool of the fake icons, double-clicking on one will attempt to load the file into the appropriate tool.

IconManager uses the following algorithm to determine whether or not to fake icons: If the icon is on the "Workbench" window, it checks the icon of the directory being scanned, and does what its flags say. If there is no icon, it fakes anyway. Otherwise, it goes by the user's selection on the "Window" menu.

1.6 Installation

Very easy - double-click on the "Install" icon and follow the instructions.
IconManager's Install uses Commodore's Installer program (not supplied).

If you want to do the installation by hand, this is what you need to do:

(1) Copy the IconManager executable and its icon to wherever you want them. Suggested place is SYS:WBStartup/

(2) Copy the doc files to wherever you want them.

(3) Copy Libs/whatis.library to LIBS:

(4) Copy S/FileTypes to S:

(5) Copy the Icons drawer to your disk.

(6) Add the following lines to S:User-Startup ->

```
Makedir RAM:Icons
Copy SYS:Icons/#? RAM:Icons ALL QUIET
Assign Icons: RAM:Icons
```

Alternatively, if you have a hard drive, you might prefer to save RAM, and leave the Icons drawer on the drive, in which case simply assign Icons: to wherever you have the icons.

To have IconManager started up every time you boot, either place it in the WBStartup drawer, or add the following line to S:User-Startup ->

```
Run >NIL: IconManager <options>
```

Note that IconManager must be in the path, unless you supply an explicit path, and that this line MUST be after the assign statement!

See
Usage from the Shell
for a description of the command-line options.

1.7 Usage from the Shell

Type "IconManager ?" at a command prompt to get the command template.
It looks like this:

```
M=MODE/N, P=SAVEPOS/S, S=SAVESTACK/S, O=SAVETOOL/S, D=SAVEDEFTOOL/S,
W=SAVEWINDOW/S, I=ICON/S, X=ICONXPOS/N, Y=ICONYPOS/N, T=ICONTEXT/M
```

The arguments act as follows:

MODE=n - sets the operation mode of IconManager, where n=0, 1 or 2 Defaults to 1

SAVEPOS - save old icon's coordinates when overwriting. Defaults FALSE.

SAVESTACK - save old icon's stacksize when overwriting. Defaults FALSE.

SAVETOOL - save old icon's tooltypes when overwriting. Defaults FALSE.

SAVEDEFTOOL - save old icon's default tool when overwriting. Defaults FALSE.

SAVEWINDOW - save old icon's window defs when overwriting (drawers and disks only...). Defaults FALSE.

ICON - create an AppIcon on the Workbench. Defaults FALSE.

ICONXPOS=n - x coordinate of the AppIcon. Defaults to NOICONPOSITION.

ICONYPOS=n - y coordinate of the AppIcon. Defaults to NOICONPOSITION.

ICONTEXT=string - text to display under the AppIcon. Defaults to "Write Icon". MUST be the last argument on the line!

IconManager operates in one of three modes. Mode is set by either the MODE switch from the Shell, the MODE tooltype from Workbench, or the MODE gadget in the GUI.

MODE=0 disables IconManager's faking of icons. The "Write Icon" function will still work, though.

MODE=1 (default) IconManager will respect the setting of "Show..." on the Workbench "Window" menu.

MODE=2 override mode. Icons will be faked regardless of the menu setting.

NOTE : When switching between modes, or switching between "Show Only Icons" and "Show All Files" on Workbench's "Window" menu, it may be neccessary to select "Update" from the "Window" menu in order for the changes to take effect.

When running, IconManager creates two AppMenuItems on the "Tools" menu. It will also optionally create an AppIcon on Workbench. This is controlled by use of the ICON switch or tooltype.

The first menuitem, "IconManager", brings up the config window when selected. If the AppIcon is present, double-clicking on it has the same effect.
See
The GUI
for details of its use.

The second menuitem, "Write Icon", will write to disk default icon(s) for any files selected when the menuitem is selected. Dropping icons on the AppIcon will also perform this action. Note that the selected icons may be real, faked or Workbench-created pseudo-icons.
When the selected object has a real icon, certain of its attributes may be preserved in the new icon. Which attributes are preserved is set by the switches above.
WriteIcon will work for any object. Disks and drawers receive the system's default icon, as do any files which IconManager cannot identify. Note that IconManager's ID routines are different to Workbench's, so that a file which shows up with Workbench's "default tool" icon may switch to the "default project" icon when WriteIcon is performed on it. Default Tool icons will only be written if the file is actually an executable.

The faked icons can be dropped onto IconEdit or similar, but they will not show up in the Shell, directory utilities (eg SID) or file-requesters. This is (currently) deliberate. If you would prefer this to be configurable, drop me a line. (See
Contacting The Author
for addresses.)

Starting a second copy of IconManager will cause the first one to quit.

WARNING : IconManager patches two library functions, dos/ExNext() and icon/GetIcon(). It does NOT check them before exiting...so anything installed after IconManager which patches the same vectors will be "cut off", and will almost certainly cause a system crash when it

is removed. If you are not sure about the state of the patches, you
can disable IconManager from its GUI. See
The GUI
for details.

Additionally, at present, IconManager MUST be run after any programs
which patch the Workbench menus, eg "ToolsDaemon" (© Nico François).
This is due to a small problem in the way in which IconManager
reads the Workbench menus...being investigated :-)
The best way to ensure this is to set the tooltype "STARTPRI=-128"
if running from WBStartup, or place it after the "LoadWB" command
in your startup-sequence if running from the Shell. If you selete
"Save" from the GUI, IconManager will write the tooltype out to its
icon automatically.

NOTE : If IconManager is launched from the Shell, it does not read its
tooltypes. Only the command-line arguments apply.

1.8 Usage from Workbench

To start IconManager, double-click on its icon. Configuration of IconManager
is by the use of tooltypes. The following are available:

MODE=n
ICON
ICONXPOS=n
ICONYPOS=n
ICONTEXT=string
SAVEPOS
SAVESTACK
SAVEDEFTOOL
SAVETOOLS
SAVEWINDOW
DONOTWAIT - needed if you are placing IconManager in WBStartup
STARTPRI=n - recommended! Set n=-128, see above for reason!

See
Usage from the Shell
for details of what they do.

When you save your configuration from the GUI, it writes new tooltypes into
the icon. These will overwrite all old ones, except the "DONOTWAIT" tooltype,
and the "STARTPRI" tooltype. Note that saving will *ALWAYS* result in the
tooltype "STARTPRI=-128" being written to the icon...

1.9 The GUI

To bring up the IconManager config window, either select "IconManager" from the "Tools" nenu on Workbench, or double-click on the AppIcon (if created).

The window is fully font-adaptive, normally to the screen font of Workbench, but if using the screen font would result in a window larger than the screen, IconManager will attempt to use the default system font, or, failing that, topaz-8. In any event, the window will adapt itself to the font when opened. It will center itself as best as possible on the mouse pointer, but it will never go off the visible part of the screen (unless it is larger than the visible part...). Should the worst occur, and come of the gadgets are not accessible, the keyboard shortcuts will still work.
Under Kickstart V39+, the checkbox gadgets will also be scaled...

The window is divided into three main areas :-

At the left side are the controls for the AppIcon:

Create Icon : If checked, the AppIcon will be created
Any Position : If checked, Workbench will put the icon in the first available position
X : If the "Any Position" gadget is not checked, you can enter the X coordinate for the AppIcon in here
Y : Enter the Y coordinate for the AppIcon here
Label : Enter the text to be placed under the AppIcon here

Notes : If "Any Position" is not checked, the "X" and "Y" gadgets will be disabled.

You can still set the coords and label for the AppIcon whether you are creating it or not, and they will still be saved to the icon.

If "Any Position" is checked, then no ICONXPOS or ICONYPOS tooltypes will be saved; if the Label is "Write Icon", no ICONTEXT tooltype will be saved. This is normal, as the defaults are "No Position" and "Write Icon".

At the right side are the settings for attribute preservation:

Position : Save old icon’s coords when overwriting
StackSize : Save old icon’s stacksize when overwriting
Default Tool : Save old icon’s default tool
Tool Types : Save old icon’s tooltypes
Window : Save old icon’s window defs (drawers/disks only)

At the bottom are the main controls:

Mode : Set the operating mode of IconManager
About... : Brings up an information window
Save : Saves the current config to IconManager’s icon

```
    Use       : Uses the current config without saving
    Cancel      : Return to the previous settings and close the
                window
    Quit        : Guess what this one does...


  Notes : If there is insufficient memory to open the config window,
    IconManager will open a requester from where you can quit
    the program if you wish.

    All gadgets in the window have keyboard shortcuts.
```

1.10 Icons:

```
                The default icons used by IconManager are stored in a drawer. The ↩
                   logical
assignment "Icons:" is made to this drawer, allowing you to put the icons
wherever you like. Floppy users will probably want to copy them to RAM:
for speed. If you have a hard drive (or low memory!), you may prefer to
leave them on your boot disk or something. Mine are in DH1:Icons/

You do not need to have an icon for every filetype IconManager can recognise.
If the file is identified, but no icon exists, then Workbench will provide
a system default icon for it. Workbench will also handle Disk/Drawer icons.
Placing "def_Disk.info" or "def_Drawer.info" in Icons: will have no effect
on this (yet...) However, the icon "def_Tool.info" is used by IconManager.
If this icon is missing, the Workbench will use its "def_Tool" icon instead.

Due to the way in which whatis.library works, IconManager differentiates
between the following types of executable : "def_Tool", "def_Pure Exe",
"def_PP40 Exe", "def_PP30 Exe", and "def_PP Exe".
This neans that even if you want them all to have the same icon, an icon must
exist for each type. See
                 WhatIs.library
                 for more information.
```

1.11 WhatIs.library

```
IconManager uses the whatis.library to identify filetypes. Whatis.library is
© Copyright S.R & P.C.

Whatis.library recognises 28 different filetypes internally, and more can be
added by means of a simple configuration file, S:FileTypes.
The types whatis.library knows about are:

  Disk
  Assign    - not used by IconManager
  Drawer
  Tool    - executable
  Pure Exe  - executable with the "pure" bit set
  PP40 Exe  - executable packed with Powerpacker V4
```

```
  PP30 Exe  - executable packed with Powerpacker V3
  PP Exe    - executable packed with Powerpacker < V3
  Script    - not sure...
  Text    - ASCII text
  Object    - object code
  Lib   - linker library
  IFF   - IFF-ILBM
  ILBM    - not sure about this one...
  ILBM24    - 24-bit ILBM
  ANIM    - IFF-ANIM
  8SVX    - IFF-8SVX
  SMUS    - IFF-SMUS
  FTXT    - IFF-FTXT
  Prefs   - IFF prefs file
  Term    - term file
  Icon    - not used by IconManager!
  Imp Data  - file packed with Imploder
  PP Data   - file packed with Powerpacker
  Zoo   - zoo archive
  LHArc   - LhA or similar archive
  MED Mod   - MED song

The default icon for each of these is "def_<name>.info" where <name> is one
of the above.
Icons for each of these (apart from Assign, Drawer and Icon!) are provided in
the "Icons" drawer of this distribution. Additionally, the S/FileTypes file
adds the follwing types to the list:

  s   - assembler source file
  i   - assembler include file
  c   - C source file
  h   - c include file
  Library   - run-time library
  Device    - device
  GIF
  JPEG
  Doc   - text file with filename including "doc" somewhere
  Readme    - text file with filename including "readme" somewhere
  Guide   - AmigaGuide file
  DMS   - dms-packed archive
  dvi   - dvi file
  tex   - tex file
  rexx    - ARexx script
  Zip   - zip archive
  pcx   - pcx graphic file
  mod   - SoundTracker/equivalent module
  imgobj    - Imagine file
  filesys   - filesystem


Icon names as above.
Icons for these are also present in "Icons".

I think all the icons are Public Domain; they've been sitting on my hard drive
for so long I can't remember where they came from now  :-)  Apologies if
they're not!
Replace them by all means, but please send me copies of any nice ones that I
```

```
can include with the next release.

To define your own types, you need to edit S:FileTypes.
Each filetype has a definition in this file. The format of these definitions
is very simple, using a few keywords to define each type:

  TYPE <string>   - starts the type entry for <string>
          Must be the first keyword.
  SUBTYPE <string>  - this type is a subtype of <string>
          This is optional.
  INSERTAFTER <string>  - places the new type after <string> in the
          filetypes list, as the list is not
          alphabetically sorted. Optional, not used by
          IconManager.
  ICONNAME <string> - <string> is the name of the default icon
          filename (minus the .info bit...).
          Optional. If not supplied, then the string
          "def_<typename>" will be used instead.
  NAMEPATTERN <string>  - <string> is a standard AmigaDOS pattern-
          matching string. The filename of the file
          being scanned must match this pattern.
          Optional, mutually exclusive with
          OPTNAMEPATTERN.
  OPTNAMEPATTERN <string> - the same as NAMEPATTERN, but may be
          overridden. Optional, mutually exclusive
          with NAMEPATTERN.
  COMPAREBYTE <n> <bytes> - test the file for <bytes> at offset <n>
          bytes in to it. Optional.
  COMPAREBYTE <n> <string>- test the file for <string> at offset <n>.
          Optional.
  SEARCHBYTE <string> - search for <string> in the first block of
          the file.
  SEARCHBYTE <bytes>  - search for <bytes> in the first block of the
          file.
  SEARCHPATTERN [CASE] <string>
        - search for <string> in file. If CASE is
          specified, search is case-sensitive.
  MATCHPATTERN [CASE] <n> <string>
        - search for <string> in file at offset <n>.
          If CASE is specified, search is case-
          sensitive.
  ENDTYPE     - ends the type entry.
  #     - anything else on this line is a comment.

 Note : <string> means ASCII text, enclosed in quotes, eg - "Hello"
  <bytes> means actual byte values, entered in hexadecimal,
  eg - $ABCD, or decimal, eg - 1234, or binary, eg - %10101
  <n> means a single number from 0 to 255 ($0 to $FF)


Example: defining the type for assembler source files:

  TYPE "asm"      ; ID string
    SUBTYPE "text"      ; the file  must also be a text file
    NAMEPATTERN "#?.(s|asm)   ; the filename must end in either ".s"
            ; or ".asm"
    ENDTYPE       ; end of type definition
```

```
Example: defining the type for IFF-ILBM files:

  TYPE "iff"      ; ID string
  COMPAREBYTE 0 "FORM"    ; check for the string "FORM" at the
          ; start of the file
  COMPAREBYTE 8 "ILBM"    ; check for the string "ILBM" at the
          ; eighth byte in
  ENDTYPE       ; end of type definition

See the FileTypes file for more examples (which work!)

Whatis.library can perform two types of scanning - LIGHT and DEEP. Light
scanning only checks the filename, so if you renamed an executable to end
in ".s", whatis.library would think it was an assembler source file if the
SUBTYPE keyword were missing. DEEP scanning overrides the filename checking,
which means that libraries (which are executables) show up as executables in
DEEP mode, but libraries in LIGHT mode.
IconManager performs a LIGHT scan, followed by a DEEP scan if the file could
not be identified in LIGHT mode, which means that libraries will show up as
libraries rather than executables...took some doing to get that working!

If any condition in the filetype definition is not satisfied, then
whatis.library will not identify the file as that type.

See the whatis.library documentation for more information.
```

1.12 Future

```
What happens next? Well, that's up to you... If people pay the shareware fee,
and send me suggestions, I will continue to develop IconManager.
Current ideas for the next release include:

  A GUI-based editor for the FileTypes file
  Support for muFS so that "protected" files don't show up at all...
  Rewrite the damn thing in C!

Any other ideas? If so, send them to me (with some money :-) and I'll see
what I can do...

If you have an idea for another program, send that along too. I'm doing a
computing course at present, so I may as well get some practise in...

After a fruitless session of trying to get "ShellMenus" to work, I am
thinking about writing my own. Any suggestions for that...?
```

1.13 Obligatory Quotes Section

```
Well, everybody's doing it these days...


    "Trust me, I'm a Doctor"  - The Doctor (c'mon BBC, bring him back!)
```

```
    "The Hedgehog Can Never Be Buggered At All"
            - Nanny Ogg (© Terry Pratchett)

    "I need a vacation"   - The Terminator

    "Mind your own business, Spock. I'm sick of your half-breed interference."
            - James T Kirk

    "UNIX soit qui mal y pense" - unknown

    "Ninety percent of everything is crud"
            - Sturgeon's Law

    "God made the integers; all else is the work of Man"
            - unknown

    "Two's company; three's a Liberal Democrat meeting"
            - unknown
```

1.14 Contacting The Author

```
This is a little complicated...

  Shareware fees should be sent to:

  Alex Taylor
  20 Nunroyd Road
  Moortown
  Leeds
  LS17 6PF
  ENGLAND


  Bug reports/suggestions/queries should be sent to the above during
  University holidays, and to one of the below in term-time:

  Internet: apt@hw.ac.uk
      ceeapt@cs.hw.ac.uk
      ceeapt@cee.hw.ac.uk

    Try them all until you get through...to say that we're not
    very well connected is an understatement!
    The first one is the one I use for anonymous ftp-ing, the
    second one is known to work from the Helios net at Aston
    University (but not from anywhere else at Aston...?), and
    the last one seems to work from Oxford...


  snail-mail: Alex Taylor
      Newbattle Abbey College
      Dalkeith
      Midlothian
      EH22 3LL
      SCOTLAND
```

Note : this address is only valid until June 1994!
After that, please use the top address if e-mail is
inappropriate or unavailable.

1.15 Acknowledgements

S.R. & P.C (whoever you are) for the whatis.library...

Commodore for making this job a damn sight harder than it should
have been! Pleeeeease make Workbench use the LVO's! That's what
they're there for!

Whoever it was wrote the program that gave me this idea...