07a22bf8-0

Jure Vrhovnik

COLLABORATORS

	TITLE : 07a22bf8-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jure Vrhovnik	January 31, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

07a22bf8-0

1.1 Dir Menu v0.9beta

Dir Menu v0.9b

This is a PUBLIC DOMAIN program.

You can use and redistribute it as long as you don't use it for commercial purpose. Please redistribute this program with all files originaly included in the archive.

This is a test (beta) version. If you find any bugs, if you want any features not yet implented, if you have any suggestions or comments, please report any thing to the author.

Click here for author's address .

1.2 About Dir Menu

Dir Menu is a pop up based file selector. You simply press a hotkey and a menu will pop up on current screen, showing all disk volumes you have. Then select a volume, and it's root directory will

appear. Then select any file on your directory, and Dir Menu will
execute selected file according to its type (if it's an IFF file
it will get shown on a screen).
But there is more! If you're still interested READ ON!
You can configure what Dir Menu should do on any type of file you will
select. You can also configure your favorite directories and files to
appear on the first display (when you press a hotkey).
But that's still not it! :-)
Dir Menu will recognize almost every known type of files. It uses
WhatIs library, and that means if there is any type of file not
already configured you can configure a new type, and Dir Menu will
remember it for ever!
AGAIN:
With Dir Menu you can instantly and quickly browse throught
directory trees on your disk drives, easily take a look at a picture
anywhere on your disk, read any text without worrying how to load a
text shower first, look into a lha archive by simply selecting that
archive file etc.

1.3 How to run Dir Menu

When you first start the program a display will pop up on a screen.
If you don't want Dir Menu to pop up on first execution you can
disable this option in the icon tool types.
If you've started the program you can now browse through your
directories by clicking on their names. When you select a file Dir
Menu will try to find its type, and will execute a command that is
defined for that type in DirMenu.cfg configuration file (if there is a
definition). If a file's type is not known, or if no action is
defined for a file type then nothing will occur. You can change that
by configuring a new file type and add a new action definition for
that type.

1.4 How to use Dir Menu

I tried to make this program to be handled as easily as
possible, while still considered enough space for future
implementations. I count on you to express any thoughs about future
versions of Dir Menu!

As already mentioned in What is this? section, you can
pop up the main display by pressing a hotkey. The default hotkey is
Left Alt + d, however you can change this to any combination
of keys you want. You may want to take a look into Tooltypes section.
When the main display is opened, all available volumes on your
computer will be displayed. (you can change this to suite your needs
in the config file) You can press on any volume
name, and a new display with root directory will appear. You can go
15 subdirectories deep.
The main display has two buttons on the top: 'Quit' and
'Close'. If you click on Quit then Dir Menu
commodity will be removed from memory. If you click on Close
button you will still be able to pop it up again by pressing a hotkey.
Every display (except from main) has the same two buttons on the
top of it: 'Root' and 'Parent'. If you click on
Parent button you will get rid of the highest display. And
clicking on Root button will take you down to the main
display.
HINT: You don't need to click on Parent button in order to
go to a parent directory: you can also click on any directory or file
on a parent display (or any display), and all higher displays will
automaticly be gone. If you select a directory, you can close a new
display by clicking on the same button again.
When you select a desired file Dir Menu will first try to recognize
what its type is. This is done with a help of whatis library
"Whatis Library" 0} (and the
FileTypes file included in this archive).If it is a file to execute it
will get executed. Otherwise it will look in DirMenu.cfg config file
for that type definition. If it finds it a defined action for that
type will be executed.
There are some predefined actions already included in Dir Menu
(such as what to do with Exe type file, or Guide - AmigaGuide file
etc. - those definitions are listed in DirMenu.cfg file , and you
can override them with your own new definitions).
If you'd like to configure new file types, please read about
WhatIs library in this manual.

1.5 Icon Tooltypes descriptions

Select a DirMenu icon and select 'icon information' from
a Workbench menu. You will be able to edit a few tooltypes, thus
changing Dir Menu's enviroment to be the most comforting to you.
Here are all tooltypes in this version:
CX_POPKEY: A hotkey that will popup the main
display when pressed.
Examples: "lalt d", "ctrl lalt p"
Default: Left Alt + d (lalt d)
CX_PRIORITY: What priority this commodity will
have in the system
environment. Usually is set to 0.
CX_POPUP: Will the main display show when you
first run Dir Menu?
Possibilities: YES, NO
If you put DirMenu in WBStartup you will set this tooltype to NO.
Default: YES
MAXITEMS: Maximum number of lines for each
display. If you set this
to 0 the only limit for maximum height of a display will be
a screen's height.
Example: 10 - only 10 lines or less will be visible.
SLIDER: This tooltype will let you change the
slider's width. This is
very usefull if you use some extra strange fonts for your screen.
0 - slider width will be the same as font's width
1 - 1.5 * font's width
2 - 2 * font's width
15 or more - number of pixels. Note that the minimum width is 15
pixels.
If you define less than this, DirMenu will force to use 15 pixels.
DIRSFIRST: Will directories appear in display
before files or after?
YES - dirs will be shown first
NO - files will be shown before dirs
Default: YES
CLOSESELECT: Will all displays close when you
select a file?

YES - close everything when selected (you can popup it again with a
hotkey)
NO - remain opened when file selected
Default: YES
SHOWTYPE: This is an advanced option for those
who want to play with
configuring new types of files, and checking if they work. On each
selected file it's action, file type, and file subtype will be
written on screen. Read about configuring new file types
"Whatis Library" 0}.
Default: NO

1.6 Configuration file description

It is not very hard to configure new actions for file types.
Firstly, a configuration file is DirMenu.cfg.
Dir Menu will first try to read it from a current directory (where Dir
Menu lies), and it is not there Dir Menu will try to read it from s:
directory. However, if it finds the config file in a current dir, it
won't try to read it from s:. If there is no config file, only
volumes will be shown in the main display, and some default actions
will be present. Those default settings will be overridden if there
is a config file.
Currently, there are only two lists to configure: Items to appear in
the main display, and Actions for specific file types.
Two commands will separate those lists in config file: definitions
after a line '#PATHS' will be used for configuring the main
display. And lines after a line '#ACTIONS' will represent actions
to be taken for certain file types.
1) Configuring main display
You can configure any path of directory or file you want to appear on
the first display. I use paths such as Picture dirs, Text file
directories etc. You can also define to have volumes or all assigns
currently defined on your system.
Main display configuration starts with a keyword #PATHS.
Next lines should be in the following format:
["]name["
text}] ["]path[
shine}"],

where "name" is a word to appear in the main display of Dir Menu,
and "path" is a full path to a file or directory you want to access.
(quotes are optional, but required if space appears in either a name
or a path).
IMPORTANT: You can define directories to act as
files or dirs. If a path ends with a slash (/), this item will be
treated as a directory, that means, if you click on it in the main
display it will show contents of this dir in a new display. If you
don’t add a slash to an end of a path Dir Menu will think this is a
file with a type "Dir", and will attempt to do an action for "Dir"
type (if you had defined it).
Examples:
#PATHS ;Starts with paths definitions
Pix hd0:pictures/ ;This will add a new entry with name ’Pix’ in the
main display. This will be a directory
Message "df0:filelist" ;This is normal text file. If you select it
Dir Menu will look for its file type, and
make an action (if it is defined)
FileDir sys:c ;This will add a directory to the main display, but
it won’t open a new display if you select it. It
will try to make an action on it if something is
defined for "Dir" file type.
2) Configuring Actions
For each type of file Dir Menu can recognize you can define any
program to get executed when a file with a certain type is selected.
For example, here are a few file types Dir Menu can recognize:
Exe - Executable file;
Text - ASCII text file;
ILBM - IFF picture;
Icon- Icon *.info file;
and many more (you should see the original DirMenu.cfg file for more
info).
To start with action configuration you must write a keyword
#ACTIONS anywhere in config file. All entries after this
should be in following format:
[']type["
text}] [']action[
shine}"]
Where "type" is name of a file type (or subtype: see WhatIs library

for more information about types), and "action" is name of a
program (with its full path) which should execute for that file type.
(quotes are again optional).
When you select a file with Dir Menu, the following procedure will occur:
- Dir Menu tries to recognize a basic type and subtype of selected
file
- It searches through the action file list for a type definition
first, and if it doesn't find the type it will search for a subtype
definition, and if not found, it will look for any default action
for that type.
- it executes a proper program (action), as if it's typed from Shell:
'action file' - where action is a defined file name,
and 'file' is selected file
VERY IMPORTANT: File types are case SENSITIVE! Keep this in mind.
If a program is not executed when it should be, check if you had wrote
a file type correctly.
TIP: If you would like to find out about file types for a
certain files on your computer you can do it by changing a tooltype
SHOWTYPE=TRUE, and restart DirMenu. For each file you select DirMenu
will tell you what is its basic type,and its secondary type (SUBTYPE).
HINT: If Dir Menu does not recognize a file type it will return
"Unknown".You are able to define a default action for any unrecognized
file (if you make a new action entry with file type "Unknown").
Examples:
#ACTIONS ;starts with action definitions
Exe "run >nil: <nil:" ;will run executables with a quiet input and output
"Text" "ppmore" ;will show text files with ppmore
ILBM "Dpaint:Dpaint" ;loads a picture into DPaint

1.7 About the future of Dir Menu

I've got a good feeling about the future of this program. I hope a
"future Commodore" will want to include it into their system. (wild
dreams of every commodity writer :-))
I really wish that you should express any comments about
this program. Click here for my address .
I have a few more ideas for future releases:
- exploit the unexploited right mouse button! However I'm not
sure right now what would be most usefull thing to do with right m.

b., although mine works better than the left one :-(
Give me your requests on this one.
- implement the way of opening output windows (every window will
open on a front screen, not only on Workbench).
- will try to do a Workbench execute emulator - if you press on
icon file it will be as if you double clicked on that icon from
Workbench. I don't know how to do that yet. If someone knows please
send me the source!
NOTE: These ideas will only be made if a usefulness of Dir Menu
is satisfactory to its users!

1.8 Thanks goes to ...

Many thanks goes to an Amiga genious, and my friend Andrej Presern
who is a real programming encyclopedia.
He helped me a lot with his advice.
I thank authors of WhatIs library: Sylvain Rougier
text} and
Pierre Carrette.
Another one goes to Ivo Kroone from Holland, who gave me an idea
that if you need a special program you simply write it, and not wait
it to get written by itself! :-)
This guide is written using Edd Dumbill's Heddley
(amiga guide creator). It's great and easy to use!
And I must not forget to thank the Amiga community for standing
by when all those horrible things about Commodore are happening
(I mean, when no things are happening) :-(

1.9 Author's address

If you have anything to say to me, PLEASE don't hesitate to
send a mail to any address written below:
email:
Jure.Vrhovnik@ijs.si
snail mail:
Jure Vrhovnik
Langusova 13
Ljubljana, 61111
Slovenia, Europe

1.10 How to install Dir Menu

You should do following steps in order to get a full working
version of Dir Menu.
1) Copy DirMenu program and this manual to any drawer you like.
2) Copy DirMenu.cfg configuration file to the
same drawer where DirMenu lies. You can also put DirMenu.cfg into
s: directory. Program will first check for this file in current
directory, and will then try to load it from s:.
3) Copy FileTypes file to your s: directory.
You MUST do this, otherwise Dir Menu won't recognize almost any
types of files!
4) Copy whatis.library to your libs: directory.
That's it. Now you can run Dir Menu without any worries.
PLEASE note that this is a beta version. I would like to
make this archive as user friendly as possible (what should be always
present for Amiga computers). You can expect an Installer script in a
full version of Dir Menu!

1.11 Defining new file types

WhatIs.library
Copyright S.R. & P.C.
[NOTE: This is an original document from WhatIs library
v4.0 authors (Sylvain Rougier and Pierre Carrette). If you have any
questions concerning WhatIs, please contact them on the address
written below:
Sylvain Rougier 39 rue Carnot 86000 Poitiers France
email: bvme@encsu1.SINet.SLB.COM (Pierre Carrette)]
[IMPORTANT: In this section you will learn that you can create new
TYPE and SUBTYPE. When DirMenu will search for actions, TYPE will be
searched first, and if not found, SUBTYPE will be searched for.]
text}
[Programmers files (sources, includes etc.) are not
included in Dir Menu archive. You can either get those from Aminet,
me, or their authors]
1.Usage
2.User's documentation
3.Programmer's documentation

1)
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 VOLU ME 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().

3.6: 25/07/93 (Internal)

Internal: Incorect scan of TagItems. Fix ed.

SEARCHPATTERN don’t work it string was not ent iled by "#?". Fixed

3.7: 12/08/93

Fixed an Enforcer hit in NextType()

Minor change: change the VERSTRING proto col.