IDer

Dave Dustin

COLLABORATORS

	TITLE : IDer	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Dave Dustin	April 18, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

IDer

1.1 IDer v1.06

```
                IDer 1.06 -- Tool launcher with Filetype identification
          Copyright © 1995-96 Dave Dustin, Eclipse Software


           ~Introduction~~~~~~~~

           ~Copyrights~~~~~~~~~~

           ~Shareware~Info~~~~~~

           ~Installation~~~~~~~~

           ~Program~Usage~~~~~~~

           ~Configuration~~~~~~~

           ~Author~Information~~

           ~Thanks~&~Greetings~~

           ~Program~History~~~~~
```

1.2 Introduction

```
IDer was written in response to my need for a program that would--when
provided with arguments--examine a file, and determine an appropriate tool
to view the file.

I downloaded one such program from Aminet, but found it severely lacking;
massively large and slow to use and configure.  I was therefore forced to
write my own.

When I started writing IDer, I decided that it should be easy to use, yet
very powerful.  And I feel I have accomplished this with the first release
```

of IDer.

IDer can recognise an unlimited number of filetypes, and uses a range of methods to identify different files. In addition, IDer can also recognise files that have been crunched.

1.3 Copyrights

Copyright Info

This program, 'IDer', is copyrighted by Dave Dustin. That means that you are NOT ALLOWED to modify the program(s) and documentation in any way. Especially you MUST NOT REMOVE the documentation.

By using this product, you accept the FULL responsibility for any damage or loss that might occur through its use or the inability to use it. The developer (Dave Dustin) can NOT be held responsible.

Distribution Info

This package is freely distributable. That means you are allowed to redistribute this package as long as you follow these points:

- Any re-distribution has to include all files in this archive, without any modifications. You are NOT allowed to add any files to the archive.

- This package may be freely distributed via BBSs; the Internet; software libraries such as Fred Fish's and Aminet CD-ROM's' and other similar electronic channels.

1.4 Shareware Info

If you use 'IDer' regularly, you are requested to send either $10, a gift of some description, or information that helps to improve IDer, such as enhancement requests or bug-reports.

This will then classify you as a 'registered user'. At the moment, this means little, but it may change in the near future.

1.5 Installation

IDer requires the following to function:

- Kickstart v37 or higher
- asl.library v38 or higher

In addition, if
xpkmaster.library
is available, then files that have been

crunched using XPK or PowerPacker can be decrunched before checking.

Simply copy IDer and IDer.info into a directory where it can be launched.

The prefs file,
IDer.prefs
, should be placed in one of the following
directories, listed in the order that they are examined looking for it:

Current Directory
S:
ENV:

Because IDer launches programs using the SystemTagList() call, they are
run as if from a Shell. In future versions I hope to offer the option to
run programs as if from Workbench.

Also, thanks to two functions by
Ralph~Babel
, IDer is able to search your
command-paths, as set by the Path shell command, when trying to launch the

ACTION
command.

1.6 Program Usage

IDer can be used via three methods, in two different environments.

~Shell~~~~~~

~Workbench~~
No matter how IDer is launched, there are several things which are
important to remember when working with certain filetypes.

- If files are crunched using either XPK or PowerPacker, IDer will
decrunch the file before it starts examination. Due to the way in which
crunched files are stored, the whole file must be decompressed, instead
of just a small portion. This does mean that with larger files,
comparison time will be slower, due to the loading/decrunch time.

The 'pre-decompression' of files can be disabled by use of the

NOCRUNCHED
option.

However, once the comparison routines are finished, the file is
completely unloaded from memory before the
ACTION
command is launched.

- Normally, IDer will wait for the
ACTION
command to finish before
carrying on to examine the next file(s).

However, some programs launch themselves asynchronously. This means that they run separately from the rest of the system, and cleanup via their own methods. (E.g. PS3M, CygnusED, etc...)

If more than one filename is selected, and some are of the same filetype, you may find several copies of the same
ACTION
command running
simultaneously. At the moment, there is nothing I can do to prevent this, but I hope to devise a solution in the future. In the mean time, be careful.

1.7 Usage - From Shell

When IDer is launched from the shell, there are three arguments available:

FILENAME - This is the name/path of the file(s) to view. Although Amiga-DOS patterns are not supported yet, multiple filenames are. Simply separate each name with a space. If there is a space embedded in the filename, simply enclose the full filename in quotes (").

If no filenames are provided, and 'asl.library v38+' is present, then a ASL File-Requester will be created, and you can select the file(s) to view.

NOCRUNCHED - If this entry is present, then usage of the XPK decrunch routines will be disabled. This means that if a file is crunched, it will not be decrunched for recognition.

CONFIG - This is the optional name for using an alternate
config
file. If for some reason you want to use a config file other than the default 'IDer.prefs', then you simply enter the full path & filename of the config file you want.

Example: CONFIG=ENV:IDer_Process.prefs

1.8 Usage - From Workbench

```
                Usage of IDer from the Workbench is similar to that of most  ↩
                   other tools
and applications.  IDer can either be used as a 'Default Tool' for a
project icon; or it can operate via normal 'Shift-Select' methods.


Normal usage would be to select the icons of the files that you wish to
view while holding down the 'Shift' key, and then double click on IDer.

Not only can normal files/icons be selected, but IDer can quite happily
handle those files using icons created by Workbench when the window is set
to 'View All' mode.


In addition to 'Shift-Selecting' files, if you simply double-click on
IDers' icon, and do _not_ have the tooltype
              APPICON
               present, then you will
be presented with a ASL file-requester, allowing you to select the files
that you wish to view.


If the
              APPICON
               tooltype is present in IDer.info, and no other files were
selected when IDer was launched, then an AppIcon will be created.

The AppIcon acts like most normal AppIcons.  If you drag a file onto the
icon, IDer will attempt to view the file.  If you double click on the
AppIcon, you will be presented with a window offering three options.

  SELECT - Creates an ASL file-requester allowing you to select files.
  QUIT   - Closes the window, and shuts down IDer.
  CANCEL - This does nothing and simply closes the window.
```

1.9 Configuration

```
                Configuration of IDer comes in two main parts


              ~ToolTypes~~~
               - These options control the operation of IDer

              ~Prefs~File~~
               - This file is the heart of IDer.  It contains all the
                   data necessary for the successful identification of
                   various filetypes.
```

1.10 Configuration - ToolTypes

```
                There are several tooltypes available for use in IDer.info.  ↩
                   These are:
```

NOCRUNCHED - If this entry is present, then usage of the XPK decrunch routines will be disabled. This means that if a file is crunched, it will not be decrunched for recognition.

CONFIG - This is the optional name for using an alternate config file. If for some reason you want to use a config file other than the default 'IDer.prefs', then you simply enter the full path & filename of the config file you want.

Example: CONFIG=ENV:IDer_Process.prefs

APPICON - This entry actually has two functions. The first, is to activate the usage of the AppIcon.

The second function of this ToolType is to specify an alternate image for the AppIcon. When the AppIcon is created normally, the image is the same as that of 'IDer.info'. However, if you provide a filename, then that will be the image used.

Example: APPICON=IDer_DropIcon.info

APPNAME - This is the text string that will appear underneath the AppIcon. The text can be upto 32 characters long. The default text is 'IDer'.

APPICONX - Specify the X coordinate of where the AppIcon will appear.

APPICONY - Specify the Y coordinate of where the AppIcon will appear.

1.11 Configuration - Prefs File

This is probably the most powerful part of IDer. It allows you to specify exactly how filetypes are to be recognised.

The prefs file is called 'IDer.prefs' and should be installed as mentioned in the installation section.

You can also provide an alternate config file by using the CONFIG

ToolType/Shell Command. This file can be called anything you like, but must be in the same format as 'IDer.prefs'

Each line in the prefs file can be either a comment, blank or a CLASS entry. Comments are identified by the line starting with either a # or ;

Each CLASS entry is a different filetype. When IDer attempts to identify what type a file is, it works it's way down the list, checking each CLASS against the file, until it gets a match.

On each CLASS line, there are eight possible entries. These are:

CLASS - This is the name of the FileType. Although it has no effect the operation of IDer, it is advisable to set it to something which you can recognise at a later date. If a space is needed in the name, please replace it with an underscore (_), or enclose the entire name inside quotes (")

Example: CLASS="ProTracker Module"

SUFFIX - Many filetypes can quickly be recognised by a trailing sequence of characters. These are called the filenames' suffix. Normally separated from the filename with a fullstop (.), this makes recognising file quick and easy.

The SUFFIX entry can be used to Identify a filetype by comparing the end of a filename with the entry. If the suffix is found, at the end of the filename, then the match is successful.

The comparison is case-insensitive, meaning it doesn't matter if it were typed in upper or lower case characters.

The method used for the comparison is equivalent to the Amiga-DOS pattern '#?<suffix>'. This means you can either include or exclude the separator character.

Example: SUFFIX="MOD"

NAME - Although most filetypes use a suffix to identify a file, sometimes some part of the filename itself will be same for every file of the same type. In addition, some filetypes place the equivalent of a suffix at the start of the filename.

The NAME entry can be used to Identify a filetype by trying to match the entry provided with the filename. The matching can make use of standard AmigaDOS patterns, allowing for greater easy of identification. For more information on DOS patterns, please see the manuals that came with your Amiga.

The comparison is case-insensitive, meaning it doesn't matter if it were typed in upper or lower case characters.

Example: NAME=(#?.LhA|#?.Lzh)

OFFSET - This is probably the most powerful of all the comparison entries. OFFSET allows you to actually compare characters _inside_ the file itself.

When the filename is specified, 2048 bytes of the file are loaded into memory for use with the OFFSET command. However, if the file is crunched, and the NOCRUNCHED entry was not provided anywhere, the whole file will be loaded into memory.

The OFFSET entry has two parts. The first is a decimal number. This is the number of bytes we should move into the file before we start comparing characters. Next comes a comma (,) that separates the "offset" from the characters to compare.

The characters to compare can be presented in two forms. They are either ASCII characters or HEX numbers.

If you specify the compare string as characters, then they must be enclosed in quotes (").

If there are characters which might change in variations on the same filetype, then the question mark (?) can be used as a wildcard. If a question mark is found in the string, then the equivalent character in the file will be skipped over during the comparison.

Example: OFFSET=0,"FORM????8SVX"

If you want to directly compare bytes out of the file, then you can specify the comparison string in HEX numbers. The string is made up of groups of two digits, one for the upper, and one for the lower part of the number. e.g. E7

If there are bytes which might change in variations on the same filetype, then two fullstop (.) characters can be used as a wildcard. If the fullstops are found in the string, then that the equivalent byte in the file will be skipped over during the comparison.

Example: OFFSET=0,E310..01

STACK - This is a decimal number specifying how much stack to set for the program when it is run. Some programs require more stack than the normal 4000 bytes.

PRI - A decimal number specifying the Priority to launch the ACTION entry at. Normally, this would be the same as IDer was when run. The number can be within the range of -127 to 127. Any numbers outside this will cause the entry to be ignored.

FLAGS - There are a number of flags that you can set for different CLASS entries to change how certain actions occur. These are:

A - If this flag is set, then ALL of the specified entries for comparison must be successful. This means that if you have a NAME, OFFSET and SUFFIX entry, then all three must match correctly, otherwise the CLASS is not match correctly.

T - If this flag is set, then at least TWO of the specified entries for comparison must be successful. This means that if you have a NAME, OFFSET and SUFFIX entry, then only one entry can fail , otherwise the CLASS is not match correctly.

I - This flag modifies the string compare routine for the OFFSET entry. If it is set, then the comparison of characters becomes CASE-INSENSITIVE. Normally, the case would make a difference and could fail the comparison.

Q - Normally when the filename is placed into the ACTION entry, the filename is enclosed in quotes ("). If this flag is set, then the filename will not be enclosed in the quotes. This is mainly for those programs which cannot parse the commandline properly.

ACTION - This is the actual commandline that will be executed when the filetype is matched correctly. The entry should not be enclosed in quotes ("), and should be the _last_ entry on the line.

Arguments for the program can be included, just as if it were a standard Shell action.

Example: ACTION=MultiView SCREEN FONTNAME=Courier.font

Normally, the filename will be attached to the end of the ACTION entry. If this position is not approprate for a certain program, it can be inserted anywhere into the commandline by inserting a '%s' where you want the filename to appear.

Example: ACTION=CygnusEdPro %s -KEEPIO

Example entry in IDer.prefs:

```
CLASS=GIF_Picture  SUFFIX=.GIF  OFFSET=0,474946  ACTION=MultiView SCREEN
```

The last entry in the prefs file should be the Default entry. This should have no entries for comparison, only the CLASS and ACTION entries and possibly FLAGS, PRI or STACK.

Thís entry will be run if none of the other entries could make a successful match.

1.12 Author Information

```
                To send bug reports, comments, suggestions, or
               shareware
                donations,
  you can contact me at one of the following addresses:


        Snail mail

                Dave Dustin
                Box 4598
                Palmerston North  5301
                New Zealand

                Phone: +64 (0)6 3579283


        EMail

                dave@eclipsnz.manawatu.gen.nz


        PGP Public keyfile

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCNAzASi/8AAAEEAOGQ9stenOQg/B1q/L7YVyw0wrbC42tyhdRXWDVrb8EZKCpV
+QWzLg9OhiZTPAmmU34VGVgFdy5ARyh8TLwWLNoCn5ntaxD/+p2eh/jWfgIjoBN3
JJZ+dKZV7jBURe5EJa+SOWchDAMELizhyw5ZOqADOO6Z9SIij92Uf4o3TtEdAAUR
tCtEYXZlIER1c3RpbiA8ZGF2ZUBlY2xpcHNuei5tYW5hd2F0dS5nZW4ubno+
=6e0L
-----END PGP PUBLIC KEY BLOCK-----
```

1.13 Thanks & Greetings

```
  Thanks must go to the following people for their contributions towards
  the successful operation of IDer:


  Betatesting

    Renze de Ruiter  <renze@trans.manawatu.gen.nz>


  Suggestions / Error reports

    Nicolas Dade  <nicolas-dade@uiuc.edu>
    Casper Gripenberg  <casper@snakemail.hut.fi>


  WorkbenchPath functions that allow for correct usage of commandpaths
  for programs launched from the workbench:
```

```
  Ralph Babel  <rbabel@babylon.pfm-mainz.de>


XPKmaster.library allowing crunched file support

  Urban Dominik Mueller  <umueller@amiga.physik.unizh.ch>


The most excellent assembler/linker package Phxass & Phxlnk

  Frank Wille  <Phoenix@axxis.owl.de>
```

1.14 Program History

```
                  Version 1.00  (5/1/1995)

    - First public release.


  Version 1.01  (6/1/1995)

    - Bugfixes
      + File locks & memory not cleared if there was a file read error
        or suitable CLASS entry could not be found.
      + Wrong error message reported under certain circumstances.


  Version 1.02  (7/1/1995)

    - Features added
      + Added
             APPNAME
              tooltype
      + Added
             CONFIG
              control entry, allowing you to select which config
        file to load and use for each copy of IDer running.


  Version 1.03  (7/1/1995)

    - Bugfixes
      + Corrected error if IDer tried to load more than file, via any
        method, in one session.  Bug was introduced in 1.01
      + Cleaned up AppIcon message handling


  Version 1.04  (8/1/1995)

    - Bugfixes
      + Corrected error with APPNAME entry.  Now displays correct text.
      + Fixed minor errors with final commandline layout.
      + Changed gadgets on AppIcon
             'About'
```

```
            requester.
    + Corrected documentation in several places.


Version 1.05  (15/1/1995)

  - No idea what changes made


Version 1.06  (18/2/1997)

  - Released to the public domain.  Ignore all mentions of Shareware
    and the like.  Source code is included in the archive if anyone
    wants to continue development.
```