ClassAction

Gasmi Salim

COLLABORATORS			
	TITLE : ClassAction		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Gasmi Salim	July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ClassAction

1.1 ClassAction 2.1 Guide

```
                        ClassAction V2.1

Read it before       ->  What is ClassAction
What do you need     ->  System Requirements

Installation         ->  How to Install
Installation Part II ->  After Install

How to use this tool ->  Using ClassAction

The Tooltypes        ->  Configuration
ClassActionPrefs     ->  Classes & Actions

Rexx Port & commands ->  The Arexx commands

What I should Add    ->  Future Improvements              Greetings
What's new & History ->  History & Features              The Author
Read This !!!!!      ->  How To Register ?                Licence
```

1.2 What is ClassAction ?

```
ClassAction is a little tool who will simplify the life of
all hard disk users....

You have a hard disk, then you have a bunch of files ...
You have executables,modules,pictures,sources,animations,sound......

ClassAction determine for you the kind of a selected file and
propose you a list of actions to do on the file.

For example if you select a GIF picture, it will be recognized
by ClassAction as a GIF Class file, and a list of actions
will be shown with Actions like 'Show' or 'Edit'

ClassAction is highly configurable, you can add classes,actions.
```

Action uses external programs, so you can use your prefered
Gif viewer to show your Gif Files (or anything else).

ClassAction has an Appicon, an Arexx port and is a commodity.

ClassAction uses xfdmaster.library to auto decrunch crunched files,
with this feature, even a crunched class can be detected.

ClassAction use very few memory and CPU time.

If you take time to configure ClassAction, you can do everything
with it !!! it's an easy way to handle files.

Before deciding to erase it, just try it !!!!!!

If you want to know more about ClassAction use More

1.3 more about ClassAction

This package is placed as ShareWare.. feel free to use it !!
and to spread it as long as you don't modify any file of the archive.
BUT if you use it regulary you must register !!!

PD Distributors are allowed to include the ClassAction Package
into their collection as long as they let me know that they have
included it.

This is release 2.1 of ClassAction ....
If you like and use it you MUST Register

I Hope you will find this prog useful.
(at least I find it useful :])

For Technicals infos about ClassAction use Technicals Infos

1.4 System Requirements

To use ClassAction , you need the following stuff :

- Amiga OS 3.0 or greater

  (May be a OS2.0 version will be released)

- a Hard Disk (ClassAction is useless without a Hard Drive)

That’s all folks !!!

1.5 Configuration of ClassAction

ClassAction interface and features are configurables via ToolTypes.

To change a Tooltype value, just select the icon of ClassAction
and select the item ’Information’ in the menu ’Icons’ from the
Workbench.

This is the list of the ToolTypes :

Nota : The default Value of each Tooltype described is used when the
associated Tooltype is not found.

==========

DONOTWAIT:

DONT remove this tooltype , it is needed if ClassAction
is loaded from WBStartup.

==========

CX_PRIORITY:

This is the Commodity priority for ClassAction.

Default Value is : 0

==========

APPSTART:

You must set this tooltype to YES (i.e: APPSTART=YES)
if you want to ClassAction start as an Appicon.

Default Value is : NO

==========

STARTDIR:

This tooltype is the directory you want to be
displayed the first time you run ClassAction.

Default Value is : Current directory

==========

WBFONT :

you must set this tooltype to YES (i.e WBFONT=YES)
if you want ClassAction use the current Workbench Font.
if you set it to YES, the real Font Width must be lower than 15,
if not ClassAction will use topaz8 font .

if you set this tooltype to NO, ClassAction will use topaz8.

Default Value is : YES

==========

DECRUNCH:

If you set this tooltype to yes (i.e DECRUNCH=YES)
ClassAction will try to decrunch crunched files
using xfdmaster library.

it's needed if you want to recognize crunched files...

Of course if you do so, the file will be pre-loaded in memory
and a buffer will be allocated to the decrunched data...
Thus, it consommes memory (around 2.5 x the file size)
So, if you don't have a lot of memory you should disable this
feature.

Default Value is : NO.

==========

AUTOSELECT:

if you set this tooltype to yes (i.e AUTOSELECT=YES)
ClassAction will check if you have only 1 action defined
for the class, if it's so it will choose it for you.

This Tooltype is only valid for the Window Part.
When ClassAction is an AppIcon it will always act as if
you turned this tooltype on.

Default Value is : NO.

==========

HEIGHT:

This tooltype is the required window height of ClassAction.

The minimum height depends of your WB font,if you supply a too
small height, ClassAction will use the smallest height possible.

Default Value is : 0 (Force to use the smallest window possible)

==========

ICONX and ICONY:

Those Tooltypes are the coordinates of the AppIcon.

But note that it's only a request, if the selected coordinates are already used by a another icon the workbench will ignore them and place the AppIcon as close as possible from your coordinates.

Default Values are : No icon position

==========

ICONNAME:

This tooltype is the AppIcon Text.
Thus if you do ICONNAME=Drop, the text displayed below the AppIcon will be 'Drop'.

Default Value is : ClassAction

==========

ICONFILE:

This tooltype is the AppIcon file.
Thus you can select your own appicon for ClassAction.
If you set this Tooltype to a icon file, ClassAction
will use it as AppIcon.
Nota: Don't supply the .info extension for the file name.

example : ICONFILE=Sys:icons/head

will use the icon head from directiry Sys:Icons/ as AppIcon.

If ClassAction fail to load your Icon file, it will use
The default one (ClassAction.info).

Default Value is : "" (Use the Default AppIcon).

==========

CLISIZE :

it is the format of cli opening window, when using SystemTags().

The syntax is : DEVICE:TopX/TopY/Width/Height/Title

thus you can select another device than CON: for the CLI's.
or define a new window position or width,height.

WARNINGS :

If you set this tooltype to a weird value, ClassAction will not
launch CLI, DONT modify it unless you know what you are doing.

** NEVER ** add device commands such AUTO, CLOSE, WAIT

```
  to this string, ClassAction will do it for you.

  ** NEVER ** put spaces into the Title.


  In Version 2.0 a similar tooltype OUTPUT was defined
  it is no more needed nor used.


  Default Value is :

    CON:0/0/640/100/ClassAction_Output_Window

=============================================================================

  DRIVE1 to DRIVE11 :

        You must set those tooltypes to a path, they will be shown
        in the buttons of the Requester.
        it's useful to quickly go to a directory.
        The syntax is:

        DRIVEx=<Button Text>,<Path>

        i.e DRIVE9=Jpeg,dh0:gfx/pictures/jpeg

        This way the button text will be 'Jpeg'
        but the path is set to dh0:gfx/pictures/jpeg.

        You can also only provide the <Button Text> without the <Path>
        i.e DRIVE3=dh0:libs

        in this case the path will be set to the same than the
        button text, but don't use too long texts, they will not be
        shown in the smalls buttons.


        Default is: NONE.
```

1.6 ClassActionPrefs program

```
        Defining new classes and actions is the heart of the program.
        To do so, you must use ClassActionPrefs program.

        Using ClassActionPrefs *should* be easy .... let's go !!



        The window is divided into two parts : Classes & Actions.

        First select a class, the associated actions will be displayed
        on the actions part.

        To add a class or delete a class, simply click on the associated
        button.
```

It’s the same concept for the actions....

What is a Class:

A class is a familly of files, for example C files can be considered as a Class , let call it C Class.

With ClassActionPrefs you can define as many classes as you want, as long as you explain how to recognize it.
to explain how to recognize a Class, you have 2 methods the matchname and the offsets.
Each Class can have as many Actions as you want.

To recognize a file , we have only 2 ways:
We can use the file name or we use file contents.

Matchname attribute is used to recognize a file regarding it’s name.
Offsets attribute is used to recognize a file with it’s contents.

You have 2 Built-in Classes that you cannot remove, they are displayed in white in the Classes ListView.

The first is Called "Unknown Class" but you can rename it if you want, this Class contains all the files that ClassAction don’t recognize.

The second one is Called "Generic Actions" and you can’t rename it, this class contains Actions that will be displayed in ALL other classes...
Why ? , if you want to have an action copy for all the classes you can create an action ’Copy’ for each class you define, but it’s looooong and boring, you can also put this action in the "Generic Action" Class , and ’Copy’ will be displayed for ALL the classes.

The generic actions are displayed in white in ClassAction Action Listview and are ONLY visible in the ClassAction Window, They will not be displayed when ClassAction is an AppIcon, or via Arexx command Load .

Creating a new Class:

Just Click on the ’add’ button on the classes part to add a class.

a class has 4 properties:

* a name
* a matchname

```
                * offsets


        the name is simply the class name, it's up to you to decide it.
        *WARNING* : a class name must be unique.


        The matchname is any regular AmigaDos expression, like :

        #?.c , mod.#? , #?.c|#?.h , #?b[a|c] , #?toto? .......

         (read the DOS manual for all wildcards)

        the matchname is not case sensitive, thus toto.C match with #?.c

 WARNING : Don't use wilcard * but use #? instead.


if you define a class using a matchname; you *MUST* be sure that
the definition is *ALWAYS* good.


Normally you shouldnt use this way to define classes.

example: if you define the GIF class with the matchname *.gif
         all files with .gif will be declared as GIF files.

         *BUT* are you sure that ALL .gif files are GIF ???

         What happens is a user rename a sound file as tree.gif ???

               Worse : if a user name a GIF file as 'picture'
                       the file wil not be recognized as a GIF.



        You should use matchname only in 2 situations :

        * matchname is a bijection of the class
          (ex: *.info is a good enough matchname for ICON class)

        * you don't have the choice ...
          (ex: how to recognize a C source, excepted with *.c)



Otherwise you *SHOULD* use Offsets.


        you can define up to 5 Offsets to define a class
        a file is declared a CLASS if *ALL* the Offsets matches.

 an offset is a place in a file where we should find something
 to recognize it.

 for example the GIF pictures always begin with string GIF.
```

```
  You have 3 syntaxes for defining offsets :

================================================================================

        Syntax #1:   Offset,HexString

================================================================================


 Offset is a DECIMAL number holding the Offset.
 HexString is an HEX string that should be found at the offset.

ex: 0,4f4a means that the file must begins with bytes $4f and $4a
ex: 9,448b3c means that at byte #9 we should find $44 and then $8b $3c


================================================================================

  Syntax #2:   Offset,'String'

================================================================================


 Offset is a DECIMAL number holding the Offset.
 String is an ASCII string that should be found at the offset.


ex: 0,'GIF'   means that the file must begins with string GIF
ex: 9,'FuBar' means that at byte #9 we should find string FuBar


================================================================================

  Syntax #3: Offset,"String"

================================================================================

 Offset is a DECIMAL number holding the Offset.
 String is an ASCII string that should be found at the offset.


 Note the difference with the previous Syntax, here we use " to define
 the string and in syntax #2 we use '.

 it's the same concept than in Syntax #2 BUT the string
 comparison is CASE INSENSITIVE.


 Then 0,'toto' will match if a file begins with ToTO .

 for example : an AmigaGuide file always begin with string : @database
 in lower or upper case.

 if you use method #2 to recognize an amigaguide file , with 0,'@database'
 ClassAction will not declare a file beginning with @DATABASE as an
```

AmigaGuide file.

it works if you define offset with method #3 : 0,"@database"

==========

if you define severals offsets for a class, of course ALL of them must matches to define the class.

ex: if the class X is defined like:

Offset #1 : 0,4a8b6c
Offset #2 : 58,14

All the files beginning with 4a8b6c AND having $14 at byte #58 will be declared as X.

To define several offets just click on Gadget 'Offset #' to activate the next offset.

Nota:

You have a Built-in Offset Command named : ASCII[]

if you put this Command into Offset #1.
i.e Offset#1=ASCII[]

This Offset will matchs with ASCII Files.
But ClassAction will try this after everything has failed.
Like this, Amigaguide files (that are ascii) will not be recognized as ascii if you have already defined an Amigaguide Class.

Normally you shoudn't use it as I have provided a standard Prefs file where the Class 'ASCII' is defined using this command.

Defining a new action :

Once the class is defined you should define actions for it.

Simply click on the 'Add' button on the action part to add an action.

an action has 5 properties:

* a name
* a run mode (CLI or WB)
* a stack size (only if run mode is CLI)
* a delay (only if run mode is CLI)
* an exec command.

the name is only the name of the action.

the run mode can be: 'CLI' , 'WB' , 'No Cli' or 'Arexx'.

**** CLI mode ****

if 'CLI' is choosen, then when selecting the action, the action will be lanched from a cli and the stack size of the cli will be determined by the stack value (default is 4096).

The run mode CLI, will only open a CLI, if it's needed,
(if the executable display something).

You can define the delay property for CLI:

if Delay is negative : (i.e Delay = -1)
then the CLI will wait until you close it by hand with the
close gadget in top left window.

if Delay is zero : (i.e Delay = 0)
then the CLI will close itself as soon as the task is terminated.

if Delay is positive : (i.e Delay = n) (n>0)
then the CLI will wait n seconds before closing itself, but
you can force the CLI closing by clicking the close gadget.

The characteristics of the used CLI can be found in
the Tooltype CLISIZE. (the old tooltype OUTPUT is obsolete now).

*** WB mode ***

if 'WB' is choosen then no cli will be opened.
and ClassAction will simulate a Workbench lanching of the action.
WARNING this mode is only valid with files that has icons.

*** NO CLI mode ***

if 'No Cli' is choosen then no cli will be opened even if the
task display something but the task is run from a CLI.

*** AREXX mode ***

if 'Arexx' is choosen then it will launch rx with the given
exec command. exec command MUST be an arexx script.
Of course RexxMaster should be Active and Rx in the directory
Sys:rexxc/ to work.

the exec command is an amigados valid command line.
YOU SHOULD always use fullpath to executables you use in this

```
    command line.
    i.e use c:copy instead of copy in exec line.


    You have buttons 'U' Up and 'D' Down to sort the actions.

    You can use button 'load' to select an executable in exec line.

With the button 'Copy' you can copy all the actions from another
class to the actual one.
Just click on copy and select the source class from where the actions
will be copied.
it's usefull when you have same actions on differents classes.


in exec lines you have arguments and commands defined by ClassAction:


    Arguments :
    -----------

    Actually 2 arguments are defined .


    [f] : fullpath of selected file
    [s] : fullpath of selected file without sufix.


    ex: let imagine you selected the file ram:main.c


    the exec line  c:copy [f] [f].bak will do :

            c:copy ram:main.c ram:main.c.bak


    the exec line c:copy [f] [s].bak will do :

            c:copy ram:main.c ram:main.bak



    example of an action :

    Action Name : Dump file
    run mode    : CLI
    Stack       : 4096
    exec line   : c:type [f]


    this will type in a cli the selected file using the executable
    c:type.


    Actually 3 request commands are defined :
    ========================================
```

```
        REQD[text] , REQF[text] and SURE[text].


        REQD[text] : Request Directory.
        REQF[text] : Request File.
  SURE[text] : Ask the user to confirm .

================================================================================

  REQs Commands


        REQs commands popup an asl requester with the title [text].
        this is useful when you want interactive command lines.
        example :

        bin:lha x [f] to REQD[Choose a Directory to unarchive]

        this will popup a directory requester letting the user to choose
        his directory target
        and the selected file [f] will be unarchived to the selected dir.

        REQF[] is the same except than it asks for a file.

        example :

        c:dir [f] > REQF[Choose a file]

================================================================================

  SURE Command:

  SURE[text] command will popup a requester with text [text]
  and with 2 buttons : yes / no.
  If the user choose no, the exec line is aborted.
  If the user chhose yes, ClassAction will execute the exec line
  on the RIGHT of the sure command.

  example :

  SURE[Really delete this file ?]C:delete [f]

  This will popup a requester asking the user to reply yes or no
  to the question "Really delete this file ?".
  if the user reply no, nothing is done; if the user reply yes
  then C:delete [f] is executed.

================================================================================


Of course you can combine any number of arguments / commands in an exec line

example :

     SURE[Really rename this file]c:rename [f] REQF[Give me a new name]
```

That's all for the moment.

1.7 Future Improvements

What I'm looking to add to this program in the next version :

- Auto Learning of new class :

i.e: you select n files you are sure they are of the same class and ClassAction will let you know how to recognize them.

- Locale.library support with French & English Catalogs.

  of course if someone wants to make foreign catalogs he is welcome, I will send him a free registered version :).

- Concept of Familly of Classes :

i.e : picture Familly contains GIF,IFF,TARGA,JPEG...
and possibility to choose a familly filter in the directory requester. Like this you can filter pictures/sounds....

- Unlimited number of path buttons.

- ClassActionPrefs use WB Fonts.

- And of course everything you'll ask me :)

1.8 About

I would like to thanks following people :

- Mireille (for her patience ....)

- Philippe Thomas (for suggestions, help and Beta testing.)

Hey Phil, if you read this, click here =-> Message for Phil

- Richier Pierre (for the MagicWB icons of ClassAction)

- Heinrich Patrick (for the Beta Testing)

- Jean Michel and Georges (for the Beta Testing on a A4000/40)

- Obvious Implementations Corp (for Dice C Pro)

- Georg Hörmann for the *GREAT* xfdmaster Library

- All registered users

1.9 The Author

You can contact me at the following address :

Gasmi Salim
6, rue des Hirondelles
F-67380 Lingolsheim
France

E-Mail: gasmisam@knossos.her.tei.gr

1.10 Using ClassAction

Using ClassAction is *REALLY* simple.

Just Select a file with the provided requester.
You can go to the Parent Directory whith the button 'Parent'
or the line <Parent> in the requester .
Right Mouse Button display the volumes & assigns.

After the selection, you will see in the right Listview
the defined actions, and the File Class.

if you reclick on the same file the first defined action
will be lanched or Just Select the action you want.....

You can Unzoom the window with the small zoom gadget in the
right top window bar.

to quit just select the 'Quit' button.

to transform the window into an appicon just close the window.
Double click on the appicon for the window popup.

That's all , Easy eh ???

1.11 Technicals Infos

ClassAction is 100% coded with DICE C 3.0

Misceleanous Infos:

The prefs file is an ASCII file named : ENVARC:ClassAction.prefs
The generic actions are saved in ENVARC:ClassAction_Gen.prefs

ClassAction create an executable called ClassAction_RunTask

```
        stored in T: , This executable is used to run WB tasks.

        The speed of the File requester is due to the sorting algorithm used
        (recursive tree sorting called 'Tri du Chat')


Libraries Information :


  ROM libraries used :

    exec.library    V37+
    dos.library   V37+
    intuition.library       V37+
    graphics.library        V37+
    gadtools.library        V39+
    workbench.library V37+
    utility.library   V39+


  DISK libraries needed :

    rexxsyslib.library  V39+
    commodities.library V37+
    asl.library         V39+
    icon.library        V37+


  DISK libraries used if found :

    datatypes.library (to recognize ASCII files)
    xfdmaster.library (to decrunch files)



How ClassAction determine a class :


        1- test if the filename matches matchnames of defined classes.

        2- test if the file matches offsets of defined classes.

        3- Decrunch the file using xfdmaster.library.

        4- test if the decrunched buffer matches offsets of defined classes.

  5- test if the file is ASCII (if ASCII Offset Command exists)


        if everything fails, the file is declared as 'Unknown Class'.
```

1.12 History

ClassAction History V 2.1 (c) Salim Gasmi

12/06/95 : V2.1

- Added the 'Generic Actions' Built-in Class.
- Added SURE[] exec command.
- Added ASCII[] Offset Command to recognize ASCII files.
- Added Arexx commands: AppIconify,Show,Status,GetClass.
- Changed running tasks system, now I use Systemtags().
  We do not need tmp files anymore.
- you can now define a delay for CLI run mode.
- Added 'string' and "string" for offsets definition.
- tooltype OUTPUT is obsolete now and not used anymore,
  we use the new tooltype CLISIZE in replacement.
- Actions requester is now well sized, appear below the
  mouse pointer and uses to frontmost public screen.
- Swapped the buttons 'Use' and 'Save" and added 'Cancel'
  in ClassActionPrefs to follow the Amiga prefs look,
  moved the button 'about' in top right corner as '?'.
- ClassActionPrefs Cycle gadgets routines weren't
  100 % system friendly and some patches like MagicCX
  makes bugs with ClassActionPrefs; It's fixed now.
- Improved the recognizer code and the Info routine:
  they are up to 400% faster.
- Listview hilight color error removed.
- ClassAction does not anymore lock the Workbench
  screen when AppIconifyed.
- A nasty bug found and removed in ClassActionPrefs.
- Right Mouse button shows Assigns only if mouse
  is in the requester and Right Mouse again brings
  back to the Directory.
- ClassAction remember now the window position.

23/05/95 : V2.00 (Major Update)

- ClassAction has now an appicon.
- ClassAction is now a commodity.

- ClassAction has now an Arexx port.

- ClassAction use now the default WB Font.

- Exec mode 'Arexx' added.

- Different color for Directories/Files.

- Up/Down gadgets added to ClassActionPrefs.

- 'Use' gadget added to ClassActionPrefs.

- Classes are now sorted into ClassActionPrefs Listview.

- APPSTART, ICONNAME, ICONX, ICONY, CX_PRIORITY,
  WBFONT, OUTPUT, ICONFILE tooltypes added.

- When the window is iconnified it have now the right height
  regarding the screen default font.

- New Save Format (CASF20).

- Suffix/Prefix Button removed. Replaced by MatchName Gadget
  who accept any Wildcard.

- Config file moved into ENVARC:

- an installer is now provided with the archive.

- some Code optimization done.

05/05/95 : V1.43

- Cleaned up the requester code, now 5% faster.

- ClassAction now look for his name using WBstartup structure
  and then can be renamed .

02/05/95 : V1.42

- 'No Cli' Exec mode added to ClassActionPrefs.

- "" are always added to filenames even if not needed
  it's easier for AREXX scripts.

06/04/95 : V 1.4

- Tooltype HEIGHT added.

- Some code optimization added.

22/02/95 : V 1.31

- If you click twice on the same file the first action will be lanched (it's faster than selected the first action by hand).

- this version is now ShareWare and you must register to get the registered version.

15/01/95 : V 1.3

- Added REQD[] and REQF[] interactive commands.

- Copy gadget added to ClassActionPrefs.

- minor improvments made.

- Beta testers reported this version is really stable.

25/11/94 : V 1.22

- First Public Release.

- Button 'Info' Added.

- ClassAction was Locking() the directories it read without UnLocking() them .... *FIXED*

- Code Optimization.

- Minor other Bugs removed.

08/11/94 : V 1.21

- Program was crashing with empty floppy units .. *FIXED*

- Volume/Name Bug Fixed

- <..> Item removed when root of a volume.

07/11/94 : V 1.2

- New interface, I have included my own fast file requester.

- STARTDIR & DRIVE1 to DRIVE11 ToolTypes added.

01/11/94 : V 1.1

- Now using xfdmaster library to recognize and decrunch files.

- Configuration with ToolTypes added (DECRUNCH,AUTOSELECT).

- 'Unknown Class' is now a built in class with unlimited actions.

- New save format (CASF11).

16/10/94 : V 1.0

- New Save Format (CASF10).

- Window has now a zoom gadget.

- a lot of classes definitions added.

10/10/94 : Beta Version

- tmp file bug removed.

- Offset increment error removed.

- using asl library for the file requester.

01/10/94 : Alpha Version

1.13 Installation

To Install this stuff :

Normally you should have the installer script with the archive
and then just use it to install it.

But if you don't, here is how to install it :

- Copy ClassAction,ClassActionPrefs and the icons where you want.

- Copy the supplied .prefs files into ENVARC:

- Copy the ClassAction.guide where you want.

That's all...

1.14 Register

If you are reading those lines, you are wondering if you
want to become a registered user of ClassAction.

Let me explain you why you should register :

First of all to support the best computer ever made,
because the future of the Amiga depends on the future software
available; then when you support a coder for his work, you support
your computer and his future !!!!

Also because I spent all my free time to try to make sharewares.
If you use them why not sending me the registration fee ??.
This will make me continue to make sharewares.

The ShareWare Version is 100% usable, nothing has been disabled,
Just for your pleasure, if you really like it, REGISTER !!

The registration fee is (regarding your local currency):

10 US$ or 20 DM or 50 FF.

Send your fee with a floppy disk to my address
And you will get the latest registered version of ClassAction.
(no annoying requesters).

Ps : Don’t forget to specify your FULL address (including country).

Thank you in advance for your support !

Yours,

Salim

1.15 Legal words

Copyright:

Warranty:

No responsibility or liability will be accepted for any damage that may
appear to have resulted from use of this program. All use is at your own
risk. The software is provided "as is" without any warranty implied or
otherwise to the fitness or accuracy of the software and documentation.
The documentation is believed to be correct but the author reserves the
right to update the software and/or documentation without notice.

1.16 The Arexx Commands

```
        ClassAction has now an Arexx port called : ClassAction.01

        This is the list of the Arexx commands supported :

=============================================================================

        Quit:

  Just quit ClassAction...

=============================================================================

        Use:

  Force ClassAction to reload the prefs file.

=============================================================================

        Ver:

  return the version of ClassAction.

=============================================================================

  Status:

  return the current status of ClassAction.

  return 0 : ClassAction is AppIconnifyed.
  return 1 : ClassAction is in Window mode.

=============================================================================

  AppIconify:

  Force ClassAction to Appiconify.

  if ClassAction is already an AppIcon, this command does nothing.

=============================================================================

  Show:

  Force ClassAction to show the main window.

  if ClassAction is already a window, this command does nothing.

=============================================================================

        Load <filename> :

        ClassAction try to load the file <filename>, and pop up
  the actions requester, letting the user choose one of them.
```

```
  if the file does not exist, this command does nothing.

==========================================================================

  GetClass <filename> :

  ClassAction try to load the file <filename> and return the
  class of the loaded file.

  if the file does not exist, this command does nothing.

==========================================================================
```

1.17 What should you do after install

```
       After the installation, you have a working ClassAction,
       with a lot of classes definitions *But* with very few Actions
       defined for them (most of the defined classes does not have any
       actions defined).

       It's up to you to define actions regarding to your system
       configuration and your prefered programs to use.

       Then just load ClassActionPrefs and configure it for your
       convenience...

       if you don't know how to configurate actions & classes, just
       read this guide :)).

       After having configurated the classes & actions , just try
       to configure ClassAction behavior with his tooltypes.

       it may take a long time to perform a 'nice' config.
       But once it's done it's really *GREAT* !!!

       Okay , now read the guide and good luck.
```

1.18 Message for Philippe THOMAS

```
  Salut Philippe !!!


  Je voulais simplement te remercier pour toute l'aide que
  tu m'as apporte a la creation de ce programme.

  Quasiment toutes les ameliorations de la version 2.0, c'est
  toi qui me les a proposes parfois meme avec insistance ..
  style le resize de la fenetre que je n'ai toujours pas fait..  :(

  Encore merci, pour tous les appels telephoniques; parfois plus
  d'une heure a discuter des Hooks, SystemTagList, de bugs etc
```

et ce meme en periode d'exams .

Franchement si ClassAction commence a etre cool , c'est beaucoup
grace a toi, je crois que j'aurais eu la flemme de le paufinner
autant si tu n'etais pas la.

Bref , ce programme est aussi un peu le tien .

Okay Phil, a la prochaine.

Salim.

Ps: Non, non ton processeur il est bien, il est pas buggé ... :)).