ClassAction

Gasmi Salim

COLLABORATORS

	TITLE : ClassAction		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Gasmi Salim	February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ClassAction

1.1 ClassAction 3.1 Guide

```
                    ClassAction
    Version 3.1
```

1.2 What is ClassAction ?

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

When you own a hard disk, you have always a bunch of files : executables, modules, pictures, sources, animations, sounds...

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

For example, when 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 'Display' or 'Edit'.

ClassAction is highly configurable, you can add your own classes and actions.
Actions use 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, is localized and is a commodity.

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

ClassAction uses 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

```
                ClassAction and ClassActionPrefs are (C) 1994-96 by Gasmi  ↩
                   Salim


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 3.1 of ClassAction.
If you like and use it you MUST
             Register
                I Hope you will find this tool 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 :


o  Amiga OS 2.0 or greater

o  MUI 3.0 or greater

o  a Hard Disk
   (ClassAction is useless without a hard disk)


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 all ToolTypes :

Nota : The default Value of each ToolType described is used when the
```

```
        associated ToolType is not found.


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

    DONOTWAIT

        DON'T remove this ToolType, it is needed if ClassAction is
        loaded from the WBStartup.

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

    DECRUNCH

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

        It is 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


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

    ICONFILE

        This ToolType is the AppIcon file.
        Thus you can select your own AppIcon for ClassAction.
        If you set this ToolType to an icon file, ClassAction will use
        it as the AppIcon image.

        Nota : Don't supply the .info extension for the file name.

        Example : ICONFILE=Sys:icons/head
                  will use the icon head.info 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 size of the CLI output 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 and dimension.


        WARNING :

        If you set this ToolType to a weird value, ClassAction will not
        launch CLI.
        DON'T 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


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

    GTLIKE

        If you simply define this tooltype , the MUI fileslector
        will look as Gadtools fileselectors ie :

        Directories in white
        Files in black
        And display .info files.

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

    CAPREFS

        This tooltype hold the path of ClassActionPrefs
        it is needed to find ClassActionPrefs when clicking
        on the prefs button.


        Default Value is : "SYS:prefs/ClassActionPrefs"

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


    DRIVE1 to DRIVE50

        Set those ToolTypes to a valid path so they will be shown
        in the buttons of the Requester.
        It is 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 #9 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 button text will be set to the same text than
        the path. But don't use too long texts, they will not be
        shown in the small buttons.


        Default Value is : "" (none)
```

1.6 ClassAction Prefs

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

Using ClassActionPrefs should be easy... so 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
buttons.


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



             1. What is a Class

             2. Creating a new Class

             3. Defining a new action

             4. The learn function
```

1.7 Using the learn function

```
The learn function is provided to help you defining new classes.

When you define a new class, you may have to define offsets for it,
and it may be long and boring to edit files to guess which offsets
```

```
could define this new class.

This is why the marvellous function learn has been made !

To use it, just follow this procedure :

1 : define your class as normal, fill name and classname

2 : click on the 'learn' button.

    and set the accuracy (75 up to 100%)
    Higher is the accuracy more reliable results will be given
    I advice you to always use 100%

3 : Select with the provided requester as many files you want
    as long as they belong to the class you want to define.
    More you select files, better will be the result.
    ClassAction will try to find out the offsets definition.
    (to select multiple files, use the shitt key.)

4: After analysis, you have a window with found offsets.

5: modify them by hand if needed.

6: click on 'Accept' to use the offsets for your class.
   or on 'Cancel' to cancel.


WARNING :

If you want the learn function works you must be sure that :

    * ALL CHOOSEN FILES BELONG TO THE SAME CLASS.

    * ALL CHOOSEN FILES ARE NOT CRUNCHED.



That's all !!! Easy no ? .
```

1.8 What is a Class

```
A class is a familly of files. For example C files can be
considered as a Class, let's 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, there are two methods :
the match name and the file contents.

Matchname attribute is used to recognize a file regarding
it's name.
Offsets attributes are used to recognize a file with it's contents.
```

There are three Built-in Classes that you cannot remove, they are displayed in white in the Classes ListView and cannot be ranamed.

The first one is called "Unknown Class"
This Class contains all the files that ClassAction cannot recognize.

The second one is called "Generic Actions"
this class contains Actions that will be displayed in ALL other classes...

Interest : if you want to have an action 'Copy' for all the classes, you can create it for each class you define, but it's looooong and boring.
A better way to do that is to create this action in the "Generic Action" Class, so 'Copy' will be displayed for all the classes.

The generic actions are displayed in white in the 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.

The third one is called "Directory" and holds actions available when you select a directory.

1.9 Creating a new Class

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

A class has 3 properties :

- a name
- a matchname
- offsets

o The name is simply the class name, it's up to you to choose it.
WARNING : a class name must be unique.

o The matchname is any regular AmigaDos expression, like :

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

(read the AmigaDOS manual for all wildcards)
WARNING : Don't use wilcard * but use #? instead.

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

If you define a class using a matchname, you must be sure that the definition is always good.

```
    Example : if you define the GIF class with the matchname #?.gif
              all files with .gif will be recognized as GIF files.

              But are you sure that ALL .gif files are GIFs, or
              that all your GIFs have the .gif extension ?

    So, you should use matchname only in two situations :

    - the 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)



o   Otherwise, you should use Offsets.

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

    For example, GIF pictures always begin with string 'GIF' at the
    offset 0.


    There are three 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 that offset.

        Example 1 : 0,4f4a   means that the file must begin with bytes
                             $4f and $4a at the position 0

        Example 2 : 9,448b3c means that at byte #9, we should find
                             $44 $8b $3c


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

    Syntax #2 :   Offset,'String'

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

        Example 1 : 0,'GIF'   means that the file must begins with
                              string 'GIF'
        Example 2 : 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 that offset.

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

        It's the same concept than in Syntax #2 but here, the string
        comparison is NOT CASE SENSITIVE.

        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 offset is defined with syntax #3 : 0,"@database"


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


    You can define up to 5 Offsets to define a class.
    A file is recognized as a class if all the Offsets matches.

    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 cycle gadget 'Offset #'
    to activate the next offset.


    Remark : The ASCII Class

    There is a built-in Offset command named : ASCII[]

    If you put this Command into Offset #1 (i.e Offset#1=ASCII[]),
    it will match with ASCII Files.
    But ClassAction will try this after everything has failed.
    Thank to 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.
```

1.10 Defining a new action

Once the class is defined you should define actions for it. Each Class can have as many Actions as you want.

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

An action has 6 properties:

- a name
- a run mode
- a stack size (only if run mode is Cli)
- a delay (only if run mode is Cli)
- an exec command
- a RescanDir flag

o Name is the name of the action.

o Run mode can be :
Cli
,
WB
,
No Cli
or
ARexx
.

o Exec command is an AmigaDOS valid command line and can contain parameters.
YOU SHOULD always use the full path for the executables.

Example : use C:Copy instead of Copy in exec line.

You can put in exec lines,
Arguments
and
Commands
built into
ClassAction.

o Set the rescanDir Checkmark if you want this action rescan the current directory , usefull for actions that change the directory contents such delete or rename a file.

Buttons

o The Up and Down arrows buttons permit to sort the actions.

o The 'Load' button asks you to select an executable in exec line.

o The 'Comm' button pops up a requester which lists all the possible

```
    arguments and commands for an exec line.

o   You can copy an existing action to the selected one by
    clicking on the gadget on the right of the Action Name
    and using the dir-tree list to select the source class
    then double click on it and then dbl click on one of
    its actions , this action will be copied to the selected one.
```

1.11 CLI mode

```
**** CLI mode ****

If 'Cli' is choosen, then when selecting the action, the action
will be launched 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 displays something).

You can define the delay property for CLI:

If Delay is negative (i.e. Delay = -1), the Cli will wait until
you close it by hand with the close gadget in the top left
of the window.

If Delay is zero (i.e. Delay = 0), the Cli will close itself as
soon as the task is terminated.

If Delay is positive (i.e. Delay = n with n>0), the Cli will
wait n seconds before closing itself, but you can force the Cli
to close by clicking the close gadget.

The dimension of the used Cli can be found in the ToolType CLISIZE.
(the old ToolType OUTPUT is now obsolete).
```

1.12 WB mode

```
*** WB mode ***

If 'WB' is choosen, then no cli will be opened, and ClassAction
will simulate a Workbench launching of the action.
The exec tool will be run with arguments specified in it's icon.

WARNING : this mode is only valid with files that have icons.
```

1.13 NO CLI mode

```
    *** NO CLI mode ***

    If 'No Cli' is choosen, no cli will be opened even if the
    program displays something, but the task is still running
    from a CLI.
```

1.14 ARexx mode

```
    *** AREXX mode ***

    If 'Arexx' is choosen, 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.
```

1.15 Arguments

```
    Currently, 8 argument commands are possible :

the first 4 commands are in lower case : [f] [s] [b] [x]
and they include the result between quotes


    [f] : full path of selected file with quotes
    [s] : full path of selected file without sufix with quotes
    [b] : filename  of selected file with quotes
    [x] : filename  of selected file without suffix with quotes


the last 4 commands does the same as before but without quotes


    [F] : full path of selected file
    [S] : full path of selected file without suffix
    [B] : filename  of selected file
    [X] : filename  of selected file without suffix


Example : let say you select the file ram:env/sys.prefs


    [f] = "ram:env/sys.prefs"
    [s] = "ram:env/sys"
    [b] = "sys.prefs"
    [x] = "sys"

    [F] = ram:env/sys.prefs
    [S] = ram:env/sys
    [B] = sys.prefs
    [X] = sys
```

```
Example : let imagine you've selected the file ram:main.c

* The exec line
             c:copy [f] [F].bak
  will be replaced by :
             c:copy "ram:main.c" ram:main.c.bak

* The exec line
             c:copy [f] [S].bak
  will be replaced by :
             c:copy "ram:main.c" ram:main.bak
```

1.16 Commands

```
Currently, five request commands are possible :


REQD[text] : Requests for a Directory.
REQF[text] : Requests for a File.
REQV[text] : Requests for a Volume.
REQT[text] : Requests for a text.
SURE[text] : Asks the user to confirm.


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

REQs Commands


    REQs commands popup a Reqtools requester with the title [text].
    This is useful when you need 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
    the target directory, and the selected file [f] will be unarchived
    to the selected directory.


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

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


    REQV[] asks the user to choose for a volume.


    REQT[] asks for a text, it's usefull to asks args for example:

        c:cpu REQT[Enter arguments for CPU]
```

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

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 choose yes, ClassAction will execute the exec line
    on the RIGHT part 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]
```

1.17 Future Improvements

What I’d like to add to this program in the next version :

- o Concept of Familly of Classes :
  For example, icture Familly contains GIF, IFF, TARGA, JPEG...
  and possibility to choose a familly filter in the directory
  requester. By this, you can filter pictures, sounds...

- o And of course everything you’ll ask me :)

1.18 About

I would like to thank the following people :

- o Mireille (for her patience...)

- o Philippe Thomas (for suggestions, help, beta testing,
  The French guide and UTT used for the install)

  Hey Phil, if you read this, click
  here

- o Bruno Durremberger (for Beta testing)
- o Jean Michel Dessolas (for beta testing on a A4000/40)
- o Obvious Implementations Corp (for Dice C Pro)
- o Nico Francois for the ReqTools library
- o Georg Hörmann for the *GREAT* xfdmaster library
- o Stefan Stuntz for the marvellous Magic User Interface .
- o All users contacted me to report bugs or for suggestions
- o All registered users

1.19 The Author

You can contact me at the following address :

Gasmi Salim
6, rue des Hirondelles

67380 Lingolsheim
France

Web: http://www.sdv.fr/pages/salim

E-Mail: salim@sdv.fr

1.20 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'

After the selection, you will see in the right listview, the
file class and the corresponding defined actions.
Just Select the action you want...

If you double-click on a file, the first defined action
will be launched.

ClassAction window is an AppWindow , thus you can throw icons
on the main window .

You can Invoque the prefs program with the 'prefs' button.

```
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.

When ClassAction is AppIconified, you can throw icons on it so the
file type will be recognised.
If the class only have one defined action, it will be executed.
Else, a window pops up with the list of all possible actions, you
just have to choose one.


Using the file manager is REALLY simple , the LEFT listview is ALWAYS
the source directory and the RIGHT one on the group "manage"
is ALWAYS the destination directory.

You have few actions with the file manager such "copy","rename" ....
and are easy to use , JUST remember that ALL file manager actions
are performed on the LEFT listview.

You can use the <--> button to swap the paths in the source/dest listviews.
.info files are not implicitly touched by the file manager .

Consider the file manager as a "bonus" and not as a real one .


That's all, Easy eh ???
```

1.21 Technicals Infos

```
ClassAction is 100 % coded with DICE C 3.0


Misceleaneous info :


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

ClassAction create an executable called ClassAction_RunTask
stored in T:
This executable is used to run WB tasks.


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+
    reqtools.library        V38+
    MUI libs                V3.0+


DISK libraries used if found :

    locale.library          V38+
    datatypes.library       V39+
    xfdmaster.library       V30+



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'.
```

1.22 Installation

```
To Install this stuff :


USE the installer script given with the archive.
To use it, just click on the install icon .

Easy eh ?
```

1.23 Register

```
                If you are reading those lines, you are wondering 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 some "annoying" requesters added: 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 to my
             address
            ,
And you will get the latest registered version of ClassAction.

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

        Thank you in advance for your support !

                        Yours,

                                Salim

1.24 Legal words

Copyright

Warranty

1.25 History

```
               ClassAction History V 3.1 (c) Salim Gasmi



09/04/96 : V3.1

    - ClassAction have now a tiny built-in File manager.

    - Some bugs removed.


12/03/96 : V3.0 (Major Update)

    - ClassAction and ClassActionPrefs now use MUI 3.0+

    - ClassAction can have up to 50 path buttons

    - ClassAction recognize directories.

    - ClassAction handle multi selection.

    - ClassAction has an appwindow.

    - You can set the Learning Accuracy now.

    - Tooltypes CAPREFS,GTLIKE added.

    - Tooltypes HEIGHT,APPSTART,WINX,WINY,ICONX,ICONY,CX_PRIORITY,CX_HOTKEY
                REQBUG,STARTDIR,WBFONT,PUBSCREEN removed and no more needed.


25/09/95 : V2.8

    - ClassAction have now a FULL commodtity support
      and a HotKey to show/hide ClassAction.

    - Tooltype APPSTART can be set to HIDE (APPSTART=HIDE)
      if you want that ClassAction start hidden.

    - Tooltype CX_HOTKEY added

    - Tooltype PUBSCREEN added to allow using Public Screens

    - The Internal File Selector of ClassAction was not freeing all
      the memory allocated, fixed now .


11/09/95 : V2.75

    - The REQD,REQF,REQV commands were not really incompatibles
      with the appicon mode, but incompatibles with some programs
      such MagicMenu (Bad luck I use MagicMenu...)

      A lot of users complained about the automatic swith into
      a REQT command even if they don't use an incompatible program.

      I have added a tooltype (REQBUG) to let the user choose to
      auto-switch into REQT or not.


03/09/95 : V2.7

    - Added commands [b],[x],[B],[X],[F],[S]

    - Added the Action arexx command.

    - removed some bugs in rendering routines

    - removed a bug with multiple icons throwed on the AppIcon.

    - The REQD,REQF,REQV commands are now swapped into REQT
      when using them from the AppIcon or Arexx , they are only
      compatible with the Window mode.


28/08/95 : V2.6

    - ClassAction window is now resizable.

    - ClassAction and ClassActionPrefs are now using ReqTools.library.

    - Added REQV command to request a volume.

    - Added REQT command to request a text.

    - Added WINX and WINY tooltypes.

    - Learn requester has now an ALL button.

    - Selected File is now in all the REQs requesters.

    - ClassAction use now a Key file for the registerd versions,
      this key file is placed in S: .


17/07/95 : V2.5

    - ClassAction and ClassActionPrefs are now localised.
      and a French catalog is provided with the archive.

    - Learn function added to ClassActionPrefs.

    - To select the first action of a file, you must now Double Click
      on it instead of reselecting the file.

    - Generic Actions are now Synchro and rescans the current directory.

    - AUTOSELECT ToolType is no more used.

    - REQs requesters opens now in current directory.
    - minor improvements made.


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 Cycle2Menu
      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 AppIconified.

    - 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
```

1.26 The Arexx Commands

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

    This is the list of all the Arexx commands :


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

    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 AppIconnified.
        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.

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

    Action <filename> <Action Pattern>


        ClassAction will execute the first matching action for the
        file <filename>.

        exemple : Action ram:toto.lha extr

        will run the first action with the name containing
        string 'extr' on the file ram:toto.lha

        The pattern is useful , you dont have to give the exact name
        of the action, just giving a part of it is enough.
```

1.27 What should you do after install

After the installation, you should 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 configure actions and classes, just read
this guide :)).

After having configured the classes and 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.

1.28 Message for Philippe THOMAS

Salut Philippe !!!

Je voulais simplement te remercier pour toute l’aide que
tu m’as apporté à la création de ce programme.

Quasiment toutes les améliorations de la version 2.0, c’est
toi qui me les a proposées, parfois même avec insistance,
style le resize de la fenêtre que je n’ai toujours pas fait.. :(

Encore merçi pour tous les appels téléphoniques ; parfois plus
d’une heure à discuter des Hooks, SystemTagList, de bugs etc...
et ce, même en periode d’exams.

Franchement si ClassAction commence à être cool, c’est beaucoup
grâce à toi, je crois que j’aurais eu la flemme de le paufinner
autant si tu n’étais pas là.

Bref, ce programme est aussi un peu le tien.

Okay Phil, à la prochaine.

Salim.

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

1.29 Known Bugs

```
ClassActionPrefs musnt be runned several times at a given time
or the 2nd instance of CAP crashes...

I’m actually searching why ... but while I’m searching do not
run 2 CAP on the same time ... anyway it is useless to run more
than one ClassActionPrefs on the same time .
```