ClipHistory

Magnus Holmgren

COLLABORATORS			
	TITLE : ClipHistory		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Magnus Holmgren	November 11, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ClipHistory

1.1 ClipHistory.guide

```
  ClipHistory 2.4

  Copyright © 1994/96-97 by Magnus Holmgren
```

Welcome to ClipHistory, a program that adds a history to the clipboard.

1.2 ClipHistory.guide/Introduction

```
  Introduction
```

This is a program that I thought about writing for a long time (a year or so! :) before I actually wrote it. I first got the idea when I had used the history feature of PowerSnap (by Nico François) a little while. I soon realised that it was a very convenient feature.

The problem is that other programs uses the clipboard as well. The history PowerSnap maintains only applies to the text that PowerSnap writes to the

clipboard, not the ones e.g. my text editor writes. And I thought that it would indeed be useful to have them in a history too.

One problem with such a history is that it easily can eat up a lot of memory, since cuts and copies in an editor can be rather large at times. To remedy this problem, there are several filtering options, that limits the amount of memory ClipHistory may use. But more about this later on in this document.

One interesting detail: ClipHistory is a completely system friendly program, in that it doesn't patch any functions at all (unless you really want it to! :). It only uses well documented features of the operating system. This does have a minor drawback: ClipHistory can miss clips, if several clips are written in a very short time. But this shouldn't be any problem, I think.. :)

1.3 ClipHistory.guide/Legal information

Legal information

ClipHistory is freeware, i.e. copyrighted, freely distributable software. Feel free to use and copy this program, as long as the following restrictions are fulfilled:

All files are copied without any alterations. If any extra files are added, it must be obvious that they don't belong to the original distribution, and that they don't need to be included in any redistribution.
Exception: So called "BBS ads" may not be added.

The copying is done on a non-commercial basis. A small fee to cover media costs etc. may be charged.

The copier isn't claiming the copyright of this program.

Any exeptions from the above requires a written permission from the author.

No Warranty

There is no warranty for the programs, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder and/or other parties provide the programs "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the programs is with you. Should the programs prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may redistribute the programs as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the programs (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties

or a failure of the programs to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

1.4 ClipHistory.guide/System requirements

System requirements

ClipHistory have very modest system requirements. It needs OS 2.04 or better, although OS 3.0 or better will make ClipHistory operate slightly better. Other than that, nothing special is needed.

If you have ReqTools (by Nico François and yours truly ;) installed, then ClipHistory will use ReqTools for its requesters (you can disable this if you like). Otherwise the "normal" system requesters will be used.

1.5 ClipHistory.guide/Installation

Installation

ClipHistory is easy to install. Simply copy the program (and its icon) to WBStartup, or whereever you find convenient. If you have OS 2.1 or higher, you might want to copy a catalog file as well, to make ClipHistory operate in another language than english.

The easiest way is to let the Installer do it. If not, copy the file cliphistory.catalog, found in the Catalogs/<language> drawer, to Locale:Catalogs/<language> (or to the drawer Catalogs/<language> in the same drawer as you placed ClipHistory). In either case, <language> should be one of the languages you have selected in the Locale preferences editor.

If there is no catalog file for your language, please try to fill in the file Empty.ct in the Catalogs drawer and send it to me. I will then include the catalog in the next release of ClipHistory. It would be nice if you could translate the Installer script as well.

1.6 ClipHistory.guide/Usage

Usage

To use ClipHistory, you first make sure that the tooltypes are set up properly. Then start it. You can also start if from a shell, but then you must remember to Run it (and to write suitable arguments). To make it permanently installed, simply drop it into WBStartup, or enter a suitable line in your S:User-Startup file.

ClipHistory installs itself as a commodity, so you can use the Commodities Exchange program to show the interface, disable/enable it, or remove it completely. Starting ClipHistory a second time (for the same clipboard

unit (option CLIPUNIT)) will cause the first copy to show the interface. Pressing the hotkey will also open the interface (option CX_POPKEY). See Options for more information.

The user interface of ClipHistory is very simple; it contains of a window with a list gadget. This list contains all the remembered clips. There is also a menu strip attached (see Menus).

The text showed in the list is either the text - if it is a text clip - or the type of the IFF clip (everything on the clipboard should be in IFF format), followed by its size (in bytes). If you have OS 3.0 or higher, the non-text clips will be shown in bold.

To select an item in the list, you either click once on it with the mouse, or you use the cursur up/down keys (optionally with one of the shift/alt/ctrl qualifiers) to highlight the requested item.

To copy an item to the clipboard, you double-click on the item, or press Return when the proper item is selected. Unless the STICKY option (see Options) have been used, the window will close.

To display a clip, you can enter Alt-Return, in addition to using the menus.

Esc closes the window.

1.7 ClipHistory.guide/Menus

```
  Menus

The window ClipHistory opens have the following menus:

Project
  About...
    Show some information about the program and the history.

  Hide
    Close the window, but don't quit.

  Quit
    Close the window and quit.

Edit
  Open clipboard...
    This will open a file requester, allowing you to select a
    file that will be written to the clipboard. If it is an IFF
    file, it will be copied as it is, otherwise it will be
    written as an IFF FTXT file.

  Save clipboard as...
    This will open a file requester, allowing you to select a
    filename to which the current clipboard contents will be
    written.

  Show clipboard...
    This will write the current clipboard contents to a
```

```
    temprary file, and start the SHOWCLIP program to show this
    file (see Options for more information),

  Clear clipboard
    Simply clear the clipboard, by writing a special (and
    small) IFF EMTY file to it.

History
  Delete item...
    This will delete the currently selected item. You can also
    press Del to do this.

  Delete all items...
    This will delete all clips in the list. You can also press
    shift-del to do this.

  Save item as...
    This allows you to save the currently selected item to a
    file of your choice.

  Show item...
    This will write the currently selected item to a temporary
    file, and start the SHOWCLIP program to show this file
    (see Options for more information),

Options
  Create icons?
    If checked, then icons will be created for the files you
    save. This is the same as the option CREATEICONS (see
    Options).

  Save text as IFF?
    If checked, then text clips (those that shows up as
    "normal" text in the window) will be saved as IFF files. If
    you save them as normal text, then some information might
    be lost (technical note: all CHRS chunks will be saved, but
    no other(s)). Same as the option TEXTASIFF (see Options).

  ReqTools?
    If checked, then ReqTools will be used for requesters, if
    it's available. Same as the option REQTOOLS (see Options).

  Sticky?
    If checked, then then ClipHistory won't close the window
    after writing a clip to the clipboard. Same as the STICKY
    option (see Options).
```

1.8 ClipHistory.guide/Options

```
  Options

Options can be specified in the tooltypes (when starting from the
Workbench) or on the command line (when starting from a Shell).

CX_PRIORITY
```

This value specifies the priority ClipHistory will have in the commodities input chain. This priority only applies to the hotkey. Default is 0. Accepted range is -128 to 127.

CX_POPKEY
This string specifies the hotkey that will open the window. Default is "control lalt c". Please refer to your Amiga manual for more information about hotkey descriptors.

CX_POPUP
If this option is set to YES, then ClipHistory will open the window upon startup (this is the default). Use NO to disable this.

CLIPUNIT
This value specifies the clipboard unit to use. Default is 0. Accepted range is 0 to 255.

You can start one copy of ClipHistory for each clipboard unit. In Commodities Exchange, the name will be "ClipHistory <unit>", where <unit> is the number you specify here.

Note: ClipHistory can be made resident. This is useful if you start several copies of ClipHistory, so that all copies share the same code (which saves quite a bit of memory).

TOOLPRI
This value specifies the "normal" priority for ClipHistory. To make sure that ClipHistory doesn't miss any clips, even if the computer is doing much work, you might want to rise the task priority of ClipHistory. Default is to use whatever priority ClipHistory was started in (this is normally 0). Accepted range is -128 to 4.

Note: The Workbench also supports the TOOLPRI tooltype. It was added to ClipHistory so that Shell users easily can change the priority as well.

WORKPRI
Some operations ClipHistory does can take a little time. Here you can specify the priority ClipHistory should use during this time, so that other programs can run normally. Usually a negative value should be entered. The default is not to change the priority. Accepted range is -128 to 4.

SILENT
If this option is set to YES, then ClipHistory won't display any requesters or flash the screen for various clip operation errors, like when there was a problem reading/writing to the clipboard, or when there wasn't enough memory to save a clip. Default is NO.

PORTNAME
This string lets you specify the name of the ARexx port ClipHistory opens. The default is "CLIPHISTORYx", where "x" is the clipboard unit the history operates on (see CLIPUNIT). If you enter an empty string (""), then no ARexx port will be opened. If ARexx isn't installed, then no ARexx port will be opened either.

NEXTKEY
  This string specifies the hotkey that will activate the next item in the history (if possible) and paste that one to the clipboard without opening the window. Useful when you have snapped a few strings and want to paste them somewere. This works even if the window should happen to be open. Default is no hotkey ("").

PREVKEY
  This string specifies a hotkey that works much like NEXTKEY, but it will paste the previous history item (if possible) instead. Default is no hotkey ("").

AUTOSELECT
  By default, NEXTKEY and PREVKEY will fail (flashing the display to inform you about it) if no clip item is selected. If this option is set to YES, then NEXTKEY will automatically select the first item in the list, and PREVKEY will select the last item.

PUBSCREEN
  This string specifies the public screen on which ClipHistory should open its window on. Default is the frontmost screen if it is public (if OPENONALL is on, it will use the frontmost screen anyway), or the default public screen.

OPENONALL
  If this option is set to YES, then ClipHistory will open on the frontmost screen, regardless if it is public or not. Default is NO.

  Note: In order for this to work properly, a function in Intuition (CloseScreen()) needs to be patched (this patch is only installed if this option have been activated). If some other program have patched the same program and you try to exit, then ClipHistory will exit anyway, but it will leave a small memory allocation, to make sure there aren't any problems. If you have a SetMan-like program installed, this should never happen.

FONTNAME
  This string specifies the name of the font (including the ".font" extension) to use for the window/menus. If not specified, then the current screen font is used. You must specify both FONTNAME and FONTSIZE.

  Hint: If you have MagicWB, it can be a good idea to use e.g. XHelvetica, since that font is "complete", i.e. chars that normally aren't printable holds graphics to identify the char in question.

FONTSIZE
  This value specifies the size of the above font. You must specify both FONTNAME and FONTSIZE.

WINDOWLEFT
  This value specifies the initial left edge of the window. Default is to center the window on the screen.

WINDOWTOP
  This value specifies the initial top edge of the window. Default is to center the window on the screen.

WINWIDTH
  This value specifies the initial width of the window. Default is
  <width of the screen> / 2.

WINHEIGHT
  This value specifies the initial height of the window. Default is
  <height of the screen> / 3.

CENTERMOUSE
  If this option is set to YES, then ClipHistory will center the
  window under the mouse. This option will override the options
  SAVEPOS, WINDOWLEFT and WINDOWTOP, if set to YES. Default is NO.

NORMALGLYPHS
  If this option is set to YES, then ClipHistory will not use custom
  symbols in the menus (only when running under OS 3.0 or higher).
  This can be useful if you e.g. use MagicMenu, which currently
  doesn't handle these symbols properly.

SCREENMENUFONT
  If this option is set to YES, then ClipHistory will use the
  current screen font for the menus, rather than the font specified
  in FONTNAME and FONTSIZE.

STICKY
  If this option is set to YES, then ClipHistory won't close the
  window after writing a clip to the clipboard. Default is NO.

SINGLECLICK
  If this option is set to YES, then a single click with the mouse on
  an item is enough to paste it to the clipboard. Click with Shift,
  Alt or Ctrl pressed to select an item without pasting (a
  doubleclick with any qualifier down will also paste).

  If set to NO, a doubleclick is needed to paste the item. A single
  click with any qualifier down will then paste the item. Default is
  NO.

POPSCREEN
  If this option is set to YES, then ClipHistory will - when closing
  its window - restore the screen that was in front before the window
  was opened. This only needs to be done if that screen wasn't
  public. Default is NO.

  Note: Using PUBSCREEN, OPENONALL or STICKY will disable this
  option.

SAVEPOS
  If this option is set to YES, then ClipHistory will remember the
  current position of the window when you close it. Otherwise it will
  reopen using the position specified in the arguments (or the
  defaults). Default is NO.

SAVESIZE
  If this option is set to YES, then ClipHistory will remember the
  current size of the window when you close it. Otherwise it will

reopen using the size specified in the arguments (or the defaults). Default is NO.

MAXCLIPS
This value specifies the maximum number of clips that the history can hold. Default is 32767. Accepted range is 0 to 32767. Note that 0 really means 1.

MAXCLIPSIZE
If the clip is larger than this value, then it will not be saved in the history. Using 0 here will disable the history. Default is the MAXMEM limit (see below) if that one is specified. Otherwise there is no limit (other than the amount of free memory).

MINCLIPSIZE
If the clip is smaller than this value, then it will not be saved in the history. The default is 0.

Please note that this value refers to the total size of the clip, including the extra overhead due to the IFF file format. For a minimal IFF file (as saved by e.g. PowerSnap), this overhead is 20 bytes. Also, an IFF file is (read: should be) always padded to an even number of bytes. Thus, a relatively useful minimum could be 22, which would force each saved clip to have at least 3 data bytes (when talking about saved text at least).

MAXMEM
This value specifies how much memory the history may use at most. The history is dynamically allocated, using so called memory pools (to reduce memory fragmentation). Thus, the actual amount of memory used might be a little more than this value. Default is no max (other than the amount of free memory).

MATCH
If this option is set to YES, check all new items against the last one, and only save the new one if it is different. Default is NO.

FULLMATCH
Like MATCH, but check all entries instead. This option overrides MATCH. Default is NO.

Note: This can take some time if you have a (very) large history (ClipHistory will use the WORKPRI during this check). If more than one clip is written to the clipboard during this time, only the last one will be remembered.

CASE
If this option is set to NO, then ClipHistory will ignore the case when comparing two clips (when either MATCH or FULLMATCH have been enavled). Obviously, this is only possible when the clip contains text (FTXT). Default is YES.

FILTER
Here you can specify which IFF types that should be saved (or not saved, if REJECT have been specified). Space separate the items. Case is important. There are many different IFF types available, and I can’t mention them all, but here is a list over the most

```
  common ones:

  FTXT
    Text. E.g. ConClip and PowerSnap saves these ones. Text
    editors are likely to write them as well.

  ILBM
    Graphics/palettes. Saved by e.g. IconEdit.

  8SVX
    Sound (samples).

REJECT
  If this option is set to YES, them the FILTER (see above) will
  reject the clip on a match instead of accept it, i.e. it reverses
  the action of the filter. Default is NO.

SHOWCLIP
  This option lets you specify the program to use to view a clip.
  ClipHistory will write the currently selected clip (or the
  contents of the clipboard) to a temporary file, and then start this
  program with the filename as the first argument. Note that this
  file will not be deleted by ClipHistory (you can do this with a
  script if you want to). Default is "MultiView".

TEMPDIR
  This option specifies were ClipHistory should write the temprary
  files that the SHOWCLIP program should read and display. Default
  is "T:".

FILETYPE
  If this option is specified, then the file type will be the second
  argument to the SHOWCLIP program. This way, a simple Shell script
  can be used to view the file (see the included ShowClip script).
  This argument will be "text" if the clip was written as a simple
  text file (TEXTASIFF is set to NO), otherwise the IFF file type
  will be the second argumet (see FILTER for more information about
  IFF types). Default is NO.

SAVEPATH
  This option lets you specify the directory in which the "Save
  clipboard..."/"Save item..." file requester should open the first
  time (it remembers the last used directory). The default is to open
  in the current directory for ClipHistory

TEXTASIFF
  If this option is set to YES then FTXT clips will be saved as an
  IFF file, rather than a normal ASCII-file. All clips that shows up
  as "normal" text in the window are FTXT clips. Note that some
  information may be lost if a clip is saved as ASCII (e.g. font
  information). Default is NO.

CREATEICONS
  If this option is set to YES, then an icon will be created for each
  clip that is saved. Default is NO.

REQTOOLS
```

```
  If this option is set to YES, then ClipHistory will use ReqTools
  for requesters, if it should be available. Default is YES.
```

1.9 ClipHistory.guide/ARexx

```
  ARexx
```

```
As of version 2.0, ClipHistory have an ARexx port. This means you can
access the history and the clipboard from your ARexx programs. The port is
called "CLIPHISTORYx" by default (where x is the clipboard unit, specified
with the CLIPUNIT argument), but you may change this by using the PORTNAME
argument. See Options for more information.
```

```
   Details

   General commands
   History commands
   Clipboard commands
   Requester commands
   Window commands

   Example scripts
```

1.10 ClipHistory.guide/Details

```
  Details
```

```
The ARexx commands in ClipHistory are documented in this manner:

  GetPrefs PREFS/A => VALUE/N

This line is then followed by an exaplanation of what the command does.
First on the line is the name of the command (in this example: GetPrefs).
Case is not importand when looking for a command. After that follows the
arguments (in some cases, there are no arguments). The => arrow indicates
that what follows after it is the return value(s) (a command may return
several pieces of information). If no => arrow is present, then the
command returns no information.

The arguments use the standard AmigaDOS template. Thus, the keyword is
optional, but must be specified if the argument doesn't come in the
specified order. The keyword is the text before the / char (and after any
, char), and the character after is a modifier, that tells you a
little about this argument. The following modifiers are used in this
document (there are others, but they are not used here):

/A
  This argument is required. An error is returned if the argument
  isn't specified.
```

```
/N
  This argument is numeric. It should be a valid decimal number, or
  an error is returned.

/S
  This argument is a switch. The precense of this keyword enables a
  flag. If it is not present, the flag is not enabled.

/F
  All text following the keyword is taken as the argument, without
  any modifications.

/M
  Any number of arguments may be specified here.
```

These modifiers may be combined, so e.g. /A/N means that a numeric value must be specified. If no modifier is present, then there are no special requirements upon the arguments (at least not anything the argument parser (ReadArgs()) can check).

These modifiers also applies to the return values. But first we need some more information about the returns. The normal way of returning information to an ARexx script is to place it in RC (limited to numeric results, since this also returns errors from the command) or RESULT (which may contain anything). In ClipHistory (and several other programs) there is a little more control. Each command that returns some information also have the two "hidden" (i.e. not documented) arguments; VAR and STEM. These arguments specify were the return value should be placed.

The VAR argument specifies in which variable the result should be placed. This is baiscally the same as to return the value in RESULT.

The STEM argument is a little different, and is mostly useful for commands that returns several pieces of information. The argument specifies the "stem base name" for the return values. Thus, if a command have the return template SIZE/N,ITEMS/N, and we specified the stem base name TEST., then the variables (stems) TEST.SIZE and TEST.ITEMS would hold the returns (numeric values). If the return is in a normal variable, then the different returns will be placed after each other, separated by spaces.

Now, there is a special return, which use the /M modifier. If we use the same stem base name as above, and the command have the return template CLIP/M, then the stem TEST.CLIP.COUNT will hold the number of strings that are returned, and TEST.CLIP.0, TEST.CLIP.1... (until TEST.CLIP.x, where x equals TEST.CLIP.COUNT - 1) will hold the different strings. If the same information is returned in a normal variable, then there will first be the count, followed by the different strings, separated by spaces.

If an argument contains spaces, then it usually needs to be "double quoted", like this:

```
  '"This is a correctly quoted string"'
```

This is needed since ARexx first removes the single quotes (') when it parses the string. ClipHistory also needs quotes, to be able to separate the arguments (these quotes must use the " char).

Negative values also needs to be quoted (e.g. '-1'). I'm not entirely sure why this is needed. Strings (e.g. file names) that contain a : char need to be quoted as well.

If an error occured, the variable RC2 will usually hold an error message or a number (a dos error code. You can use Fault to get an explanation for a certain number). Note that "OPTIONS RESULTS" must have be used in the script for this to happen.

All commands sets RC to 10 for an error, 5 for a warning (e.g. asking for a clip item that doesn't exist, or trying to do a window operation when the window isn't open), and 0 for all ok. The error code 20 is reserved for fatal errors, making ClipHistory unable to parse the command, or properly return the result. The typical reason for this to happen is lack of memory. This is also returned if there are any pending commands when ClipHistory is about to exit.

The ARexx interface in ClipHistory was originally developed using ARexxBox. The generated code was then modified in several ways to get it smaller (without loosing any functionality).

1.11 ClipHistory.guide/General commands

```
  General commands

About
  Simply open the about window. It will open on the same screen as
  the main window, or on the default public screen.

CxActivate
  Activate the Commodities interface. Similar to the "Active" gadget
  in the Exchange program. Returns 5 if the interface
  already was active.

CxInactivate
  Similar to CxActivate, but inactivate the interface instead.
  Returns 5 if the interface already was inactive.

GetPrefs PREFS/A => VALUE/N
  Get the state of the specified preferences item. Currently, the
  following items may be specified:

  ICONS
    The "Create icons?" menu item (CREATEICONS argument).

  REQTOOLS
    The "ReqTools?" menu item (REQTOOLS argument).

  TEXTASIFF
    The "Text as IFF?" menu item (TEXTAIFF argument).

  STICKY
    The "Sticky?" menu item (STICKY argument).

  This corresponds to what is shown in the Options menu. See the
```

```
  Options section for more information about these options.

  Currently, the value is set to 1 if the flag is set, or 0 if the
  flag isn't set.

LockGUI
  Lock the GUI, so that no input can be made (output can still me
  made). These calls nest, so remember to match them properly.

Quit
  Quit ClipHistory. No confirmation requester will be shown.

Show PUBSCREEN
  Open the GUI. If a PUBSCREEN argument is specified, then
  ClipHistory will try to open on the specified public screen.

UnlockGUI
  Decrease the nesting count for the GUI lock. If we reach zero,
  unlock the GUI, re-enabling user input.

  If you call this command too many times (making the count
  negative), the count is reset to zero, and return 10.

SetPrefs PREFS/A,VALUE/A/N
  Set the state of the specified preferences item. Currently, the
  following items may be specified:

  ICONS
    The "Create icons?" menu item (CREATEICONS argument).

  REQTOOLS
    The "ReqTools?" menu item (REQTOOLS argument).

  TEXTASIFF
    The "Text as IFF?" menu item (TEXTAIFF argument).

  STICKY
    The "Sticky?" menu item (STICKY argument).

  This corresponds to what is shown in the Options menu. See the
  Options section for more information about these options.

  If the value is 1 (or rather, not 0), then the flag is set, or if
  it is 0, the flag is cleared.
```

1.12 ClipHistory.guide/History commands

```
  History commands

DeleteAllItems FORCE/S
  Delete all items in the history. Use FORCE to disable the
  confirmation requester. Returns 5 if the user cancelled the delete.

DeleteItem ITEM/N,FORCE/S
  Delete the specified item, or the currently selected if none was
```

```
  specified.  Use FORCE to disable the confirmation requester.
  Returns 5 if the user cancelled the requester, or the specified
  item didn't exist.

GetTextItem ITEM/N => CLIP/M
  Return the contents of the specified item, of if not specified, the
  currently selected item. If the clip doesn't contain any text,
  return 5 in RC. The CLIP array will hold the different CHRS
  chunks. Usually, there is only one.

GetSelected => ITEM/N
  Returns the ordinal number of the currently selected item, or -1 if
  no item is selected.

ItemInfo ITEM/N => TYPE,SIZE/N,NUMCHRS/N
  Return some information about the specified item, or the currently
  selected item if no argument is given. The follwing information is
  returned:

  TYPE
    The type of the IFF file, usually FTXT, ILBM or 8SVX.

  SIZE/N
    The size of the clip, in bytes.

  NUMCHRS/N
    The number of so called CHRS chunks in the clip. If 0, then
    the clip doesn't contain any text.

HistoryInfo => ITEMS/N,SIZE/N
  Returns the following information about the history:

  ITEMS/N
    Number of items currently in the history

  SIZE/N
    The total size of the history, in bytes.

LockHistory
  Lock the history, so that no items can be added to it. These calls
  nest, so remember to match them properly.

PutItem ITEM/N
  Write the specified or currently selected item to the clipboard.

SaveItemAs NAME,ITEM/N,REQUEST/S
  Save the specified or currently selected item to the specified
  file. If no file name is specified, or the REQUEST flag is used,
  show a requester to let the user specify the name. NAME then
  specifies the default name in the requester.

SelectItem ITEM/N,RELATIVE/S
  Select the specified item. If the RELATIVE flag is used, add the
  ITEM to the currently selected item number (there must be a
  selected item for this to work).

ShowItem ITEM/N
```

```
  Show the specified, or currently selected item. See SHOWCLIP in
  the Options chapter for more information about this.

UnlockHistory
  Decrease the nesting count for the history lock. If we reach zero,
  unlock the history, making it possible to add new items.

  If you call this command too many times (making the count
  negative), the count is reset to zero, and return 10.
```

1.13 ClipHistory.guide/Clipboard commands

```
  Clipboard commands

ClearClipboard
  Clear the clipboard, by writing a special "empty" IFF-file to it.

GetTextClip => CLIP/M
  Return the contents of the clipboard. If the clipboard doesn't
  contain any text, return 5 in RC. The CLIP array will hold the
  CHRS chunks. Usually, there will only be one.

PutFile NAME,REQUEST/S
  Place the named file on the clipboard. If no name is given, or the
  REQUEST flag is set, open a requester for the user to select a
  file. NAME then specifies the default name in the requester.

  If the selected file is an IFF file, it will be copied as-is to the
  clipboard.  Otherwise, ClipHistory assumes the file contains text,
  and will write it as an IFF FTXT file.

PutText TEXT/A/F
  Put the specified text on the clipboard, as an IFF FTXT file.

SaveClipAs NAME,ITEM/N,REQUEST/S
  Save the clipboard contents to the specified file. If no file name
  is specified, or the REQUEST flag is used, show a requester to let
  the user specify the name. NAME then specifies the default name in
  the requester.

ShowClip
  Show the file in the clipboard. See the SHOWCLIP argument in the
  Options chapter for more information about this.
```

1.14 ClipHistory.guide/Requester commands

```
  Requester commands

RequestFile TITLE,PATH,FILE,PATTERN,SAVEMODE/S => FILE
  Show a file requester, with the specified title, default path and
  file, and the specified pattern, and return the selected name
  (FILE). Use the SAVEMODE switch if you request a file to write
```

```
  data to. If the requester is cancelled, return 5 in RC.

RequestNotify PROMPT/A
  Show the specified prompt in a requester with a single "Ok" gadget.

RequestResponse TITLE,PROMPT/A,GADGETS
  Show the specified prompt, with the specified title in a requester
  with an "Ok" and a "Cancel" gadget. Returns 5 if cancel was
  selected, or 0 if ok was selected.

  The GADGETS option allows you to specify your own gadgets. The
  different gadgets should be separated by "|" chars. Returns 0 for
  the rightmost gadget, 1 for the leftmost, 2 for the next, and so
  on...
```

1.15 ClipHistory.guide/Window commands

```
  Window commands

ActivateWindow
  Tries to activates the main window, if open. Returns 5 if the
  window isn't open.

ChangeWindow LEFTEDGE/N,TOPEDGE/N,WIDTH/N,HEIGHT/N
  Move and resize the window to the given coordinates, or as close as
  possible. '-1' for a given coordinate means "don't change this one".
  Returns 5 if the window isn't open.

Hide
  Simply close the window.

IsOpen => OPEN/N
  Returns 1 if the window is open, or 0 if it isn't.

MoveWindow LEFTEDGE/N,TOPEDGE/N
  Similar to ChangeWindow, but this command only lets you change the
  position of the window.

SizeWindow WIDTH/N,HEIGHT/N
  Similar to ChangeWindow, but this command only lets you change the
  size of the window.

UnZoomWindow
  If the window is zoomed, then unzoom it (possibly making it
  larger). Returns 5 if the window isn't open.

WindowToBack
  Simply move the window behind any other window(s), if possible.
  Returns 5 if the window isn't open.

WindowToFront
  Simply move the window in front of any other window(s), if possible.
  Returns 5 if the window isn't open.

ZoomWindow
```

```
  If the window isn't zoomed, then zoom it (possibly making it
  smaller). Returns 5 if the window isn't open.
```

1.16 ClipHistory.guide/Example scripts

```
  Example scripts

Thogether with ClipHistory, you should find a couple of example ARexx scripts:

Test.cliph
  A simple test script, that traverses all clips in the history, and
  collects some simple statistics about it.

Dump.cliph
  Writes the contents of the history to a drawer, using a special
  name pattern, namely ClipHistory.<n>  (where <n> is a decimal
  number). Any previous files matching that pattern will be deleted.
  Edit the script to change the drawer the clips are written to.

Restore.cliph
  The reverse of Dump.cliph. That is, it loads all clips (matching
  the above pattern) found in the drawer (specified in the script)
  into ClipHistory, optionally deleting the previous history
  contents.

The two last scripts are included with ClipHistory 2.4 and onwards. They
allow you to easily make a snapshot of the current history, for reload at a
later date.
```

1.17 ClipHistory.guide/Future

```
  The Future

There are a few things I might add to ClipHistory:

  More IFF parsing. Now it is minimal, only reading/writing exactly
  what is on the clipboard, with a minor "support" for FTXT.
  Enhanced parsing could e.g. reduce the overhead for FTXT clips, by
  only saving the actual text.

  Load/save of the entire history. Perhaps automated, so that the
  history contents can be (automatically) restored after a reboot.

  Store large clips on disk rather than in memory.

  Show (scaled) versions of graphics clips somehow. This would
  require OS 3.0 for several reasons.

  Prefs window, to edit/save the settings. Perhaps as a separate
  prefs program. I have done some work in this direction, but I've
  never finished it...
```

```
  Some other way of "highlighting" non-text clips on OS 3.0 or
  better. The problem is that I don't know how I should change it. :)
  Please tell me if you have any idea.

  AppWindow support. Dropped icons should then be pasted to the
  clipboard.

  Split and join clips (at line feed chars)

These things are not that important for me, but feel free to send me a note
telling me what you would like to see added (see Author contact).
```

1.18 ClipHistory.guide/Author contact

```
  Author contact

Feel free to send comments, bug reports (detailed ones, please),
translations, money or whatever, to:

  S-mail:
    Magnus Holmgren
    Kvarnbergsvägen 5
    S-444 47  Stenungsund
    SWEDEN

  E-mail:
    cmh@lls.se
    2:203/512.10@fidonet.org
```

1.19 ClipHistory.guide/Acknowledgements

```
  Acknowledgements

Thanks go to the following persons, for helping me in some way with this
program:

Nico François
  His program PowerSnap inspired me do write this program in the
  first place. He also sent me a few small (but useful) functions,
  and came with a few suggestions/bug reports.

Nikolai Waldman
  Suggestions.

Michael Berg
  Bug report and suggestions.

Lars Eilebrecht
  German translation.

Georges Concalves
  French translation and MagicWB icon.
```

```
Kenneth Fribert
  Danish translation.

Michael Balzer
  For ARexxBox, which was used as a base for the ARexx interface
  (the code have been modified a "little".. Saved ~5 Kb from the
  program! :).

Olaf Barthel
  For the "menu glyph" code from term.
```

1.20 ClipHistory.guide/Program history

```
  Program history

Version 1.0:
  Initial release.

Version 1.1:
  Removed some debugging code.

  Added the SINGLECLICK flag. If YES, a single click on an item is
  enough to paste it to the clipboard. Use shift/alt/ctrl to select an item
  without pasting (a doubleclick with any qualifier down will also paste). If
  set to NO, a single click with any qualifier down will also paste the item.

  Recompiled with DICE 3.0.

  German catalog added. Unfortunately I don't have the source for it.. :)

Version 2.0:
  ARexx-port with over 40 commands added.

  ClipHistory is now residentable! It wasn't that easy... :) (try
  yourself with DICE, a couple of callback hooks and a patch! :)

  French catalog and installer script by Georges Goncalves included.

  Danish catalog by Kenneth Fribert included.

  German catalog source included.

  MagicWB icon by Georges Concalves included.

  Added menu items to load/save/view/clear the current clipboard
  contents. You can also view an item in the history.

  Any font specified in the tooltypes needed to be in memory for
  ClipHistory to find it. Now it will try to load it from disk if
  possible.

  If the screenfont or the font specified in the tooltypes didn't
  open, ClipHistory will now use the system default font instead
  (rather than topaz/8).
```

Added the following arguments: PORTNAME, NEXTKEY, PREVKEY, AUTOSELECT, SHOWCLIP, TEMPDIR, FILETYPE, and SAVEPATH. See Options.

ClipHistory will now always exit even if someone have overpatched the function ClipHistory may patch. If the patch was overpatched, there will be a small memory loss.

When doing a full check, and a match is found, then ClipHistory will select the item that matched, so that the selected item reflects the current clipboard contents. In general, ClipHistory will do its best to keep the selected item up to date. This includes deselecting any item if no item in the history matches the current clipboard contents (e.g. a paste was filtered out)

ClipHistory will now use the OS 3.0 wait pointer when needed.

SysIHack "support" added (i.e. it handles larger-than-normal window size gadgets properly).

The window will now backfill with the current background color, rather than color 0.

The "Settings/Create icons" menu item was not working, in that icons never got created. I had written a function to do it, but it wasn't used anywere! :)

Non-FTXT clips are now shown in real bold (if possible).

When deleting a single item, a new item will be selected, if possible.

Version 2.1:

The selected item wasn't always visible when the window opened.

Removed Enforcer hits when data was written to the clipboard and the window wasn't open.

Version 2.2:

When copying a non-text clip from the clipboard to a file, then ClipHistory would read innocent memory, and generate an incorrect (and often a very large) file.

More tweeking of the ARexx code. Made it somewhat smaller, and changed some other code to be more tolerant (e.g., stem base names no longer require a '.' char at the end. If not present, it will be added).

Made a few minor fixes to the example ARexx-script.

Added scaled menu glyphs (OS 3.0+ only), based on source code from term.

Date in About requester localized. :)

The ReqTools file requester didn't open in the right drawer.

```
Version 2.3:
  ARexx-commands that return information caused Enforcer hits, if no
  STEM-name was specified.

  Fixed another bug in the ARexx-interface, that could cause
  incorrect returns. I'm not sure if it actually could happen with
  the current command set, but... ;)

  Recompiled with SAS/C 6.56. Shaved off a few KBs.

  Alt-Return can be used to show the current clip in the GUI.

  ClipHistory didn't make items visible properly on pre-OS 3.0
  Amigas.

  The ARexx-command PutText didn't always return errors as it
  should.

  If MATCH was specified, and a newly added clip was identical to
  the last, the last clip was either not activated, or it was
  deactivated. Now it is properly activated.

  Added some new options: CENTERMOUSE, NORMALGLYPHS, SCREENMENUFONT.
  See Options for more information.

  Added the STICKY option to the Options menu.

  Stopped using TexInfo for this manual. This means that the
  ASCII-version will no longer be included.

  Specifying PORTNAME="" didn't stop ClipHistory from creating an
  ARexx-port with the default name.

  NewIcons included. From various sources, with minor changes by me.

  Fixed some bugs in the cleanup code, that could cause problems if
  ClipHistory failed to start up properly.

  Cleaned up some code.

Version 2.4:
  Invalid clips (to be precise, clips with an odd "file" size) were
  saved into the history when they should be rejected. Also, they
  were saved in an incorrect state, causing problems when accessed
  later.

  Recompiled with SAS/C 6.57.

  Added SAVEMODE switch to the REQUESTFILE ARexx command.

  Included some more example ARexx scripts, that actually can be
  useful, namely Dump.cliph and Restore.cliph. Dump.cliph writes
  the contents of the history into a drawer (specified within the
  script), using the pattern ClipHistory.<n> (where <n> is a decimal
  number). Any previous files with such a name will be deleted.
  Restore.cliph can then be used to reload that saved history set
```

(optionally replacing the previous history contents).

Fixed an error in the documentation to the ARexx command RequestNotify. The return value is a warning (5) if "Cancel" was selected (and no GADGETS specified), and not the other way around.

A couple of translations added and updated, done by ATO.