SysInspector

COLLABORATORS			
	TITLE : SysInspector		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SysInspector

1.1 SysInspector - Documentation

SysInspector 1.4
"Your Amiga can't hide it from the Inspector ;)"

SHAREWARE.

1.2 About

There are already quite a few system monitors available on Aminet.
Unfortunately they all suffer in at least one of these areas:

\textdegree{} They are old and outdated, not supporting any OS 3.x feature like
memory handlers.

```
 \textdegree{} They are buggy or very unstable.

 \textdegree{} They have a lousy GUI, sometimes not even fully font-sensitive, or
   looking plainly ugly under modern system configurations.

 \textdegree{} One lacks a feature only available in the other, and vice-versa.

 \textdegree{} They don't even come with english documentation (!!!), so you're
   pretty much left "on your own" while using it (and considering how
   easy it is to crash your system when misusing a system monitor, it
   often leads to disaster).


So there's SysInspector.  The main goals while developping it were:

 1) Provide something STABLE.  Of course, removing a task is always risky.
    But there are some system monitors out there who would crash just at
    trying to give you the list of tasks.  Not very handy when your
    intention is to get rid of an already crashed task, not to crash
    another one.  To help ensuring this, SysInspector has its own guru
    trapper built-in, which should catch most software failures from
    itself.  It won't help you though if you crash another task using
    SysInspector.

 2) While making it quite powerful and handy for the advanced developper,
    still keep it simple enough for the casual user who would only want to
    change a task priority, or close the window left open by a crashed
    program.  Some features are best fit into another tool such as
    HDToolbox or your usual debugger than into a system monitor.

 3) Provide it an attractive, font-sensitive and easy to use GUI, without
    necessarily turning it into a snail nor a resource-hogger.

 4) Make it actualy USEFUL.  Some monitors let you see what's on your
    system, might even let you do a few funky things...  but nothing
    useful.  What's the use of being able to remove a crashed task if it
    won't also remove that annoying window cluttering your Workbench?

 5) Provide a minimum of configurability.


WARNING!

Careless use of a system monitor can lead to crashes, even possible loss
of data.  PLEASE take the time to read this documentation!  The risky
operations are labeled as such, so you'll know what can be safely done
and what cannot.
```

1.3 About ClassAct

```
 [Taken from the ClassAct homepage]

ClassAct is a set of BOOPSI classes co-authored by Christopher Aldi,
Timothy Aston, Osma Ahvenlampi and Petter Nilsen, published by Amigability.
```

ClassAct provides object-oriented building blocks for your application in the form of Intuition BOOPSI classes libraries libraries. As they are standard classes, they may be used with any application environment supporting BOOPSI. ClassAct is a complete GUI system in its own right, supporting everything from simple buttons to an advanced list management class, and includes a complete GUI layout gadget class, arexx class and window class.

ClassAct user interfaces are font-adaptive, and dynamically resizable. Like any modern GUI, the ClassAct layout engine does not use fixed coordinates, but instead hierarchical groupings. The layout engine supports not only gadgets within its group, but images and even other layout groups as well.

Programs that use ClassAct can be made freely distributable, shareware, commercial, etc. There is no fee for your users!. When you purchase the ClassAct development kit license, users of your software get to use all the functions of our classes. ClassAct is not only a powerful and time-saving choice for software developers, but an affordable and convenient one as well.

ClassAct is an expanding project, providing you with all the graphical user interface tools you need to write your application. Currently there are over 25 different classes, and this list is growing all the time!

ClassAct requires at least Workbench 2.04, and will take advantage of higher OS versions if available.

ClassAct development kit supports SAS/C, Dice, and AmigaE.

Latest version of the classes can be found on Internet at:

ftp://ftp.thule.no/pub/classact/

1.4 Usage

To start SysInspector, you can double-click on the icon, or start it from a shell. Command template is:

```
DISPLAY/K/N, ICONIFY/S, TASKPRI/N/K, PUBSCREEN
```

DISPLAY - Select an initial display type (bypassing prefs). Must be one of these values:

0 - Assigns
1 - Commodities
2 - Devices
3 - Fonts
4 - Interrupt handlers/servers
5 - Libraries
6 - Filesystem locks
7 - Memory nodes
8 - Memory Handlers

```
           9 - Message ports
          10 - Mounted devices
          11 - Resident modules
          12 - Resources
          13 - Screenmodes
          14 - Screens and Windows
          15 - Signal Semaphores
          16 - System Information (default)
          17 - Tasks and Processes
          18 - Timer.device Requests
          19 - Resident DOS Commands


ICONIFY - Start SysInspector in an iconified (or menufied) state.


PUBSCREEN - Name of the public screen on which SysInspector should open.

TASKPRI - Start SysInspector with a different task priority.  Especialy
          useful if you want to start it while some other (buggy) task
          is eating up most of the CPU time.

          NOTE: Tho there's a TASKPRI tooltype, it's strongly recommended
                to only use this argument from the CLI, since the args
                parsing in that case is being done earlier (it will be
                done before opening any gadget/classes), so you will
                benefit from the new task pri during this part of the
                startup code as well.


   Those same arguments can also be specified as tooltypes, which will be
used if you start SysInspector from the Workbench.
```

1.5 Requirements

```
To run, SysInspector requires:


  \textdegree{} Any Amiga model, with a 68000 or better.

  \textdegree{} Kickstart 3.0 or better.  You must also have SetPatch installed,
    so that memory pools properly works.

  \textdegree{} ClassAct 2.0 or better (the latest version can be found on  ↩
     Internet
    at ftp://ftp.thule.no/pub/classact/).  Note that bugs present in
    earlier releases of ClassAct could cause SysInspector to malfunction -
    make sure you do have the latest version (some minimal revisions are
    enforced to make sure you're not using one of the gadget versions
    known to cause problems with SysInspector).
```

1.6 Installation

Just double-click on the supplied Installer script. If you prefer to do it manualy, copy the supplied libraries in your own libs: drawer (check first their version!), and copy the main executable wherever you wish.

If you haven't installed ClassAct yet, do so through its own Installer script.

The explorer.config file must me copied in your S: assign.

1.7 Main Window

At the top of the window, there's a chooser gadget (or an array of buttons, depending if you enabled "Use Button Array" in the prefs or not) that let you select the information you wish to display in the list (the "list type"). If you have the Button Array enabled, the corresponding list type will be shown in the window's titlebar while pressing on the button. You can also see what a button does by enabling the "Button Array Popup Help" either from the preferences or by pressing on the "Help" key, and leaving the pointer over a button.

If you can't see the complete button array in the window, you can scroll the array by holding down shift, and dragging the button array with the left mouse button pressed.

Just below there's a listbrowser that displays the information of the selected list type. In most cases, this list will be separated in columns, depending on the selected list type. You can navigate through it by using the usual arrows/scrollbar located around it, or by using the keyboard:

```
     Up/Down Cursor keys: Move up/down by one entry
   Shift Up/Down Cursors: Move up/down by one page
    Ctrl Up/Down Cursors: Move to top/bottom of the list

  Left/Right Cursor keys: Change the displayed list type
```

For some list types (screens/windows, Interrupts, Locks), the list will be displayed in an hierarchical format. Nodes that have childs will display a small arrow pictogram to their left. Just click on the pictogram to expand the list and display the childs for that given node, or to hide them. An example is the Screens/Windows list: when you select this type, a list of the currently opened screens will be shown To view the windows opened on a specific screen (if there's any), just click on the arrow located to the left of the node.

At the bottom, there's two rows of buttons, and a text gadget displaying the current status. Not all of these buttons are used for each types: once you select a node in the list, the appropriate buttons will be enabled. Their use is documented in the list type sections of this manual.

Two buttons are special cases: "More", and "Refresh".

```
   Refresh: Will rescan your system, to update the current list.
```

So, if you were displaying tasks, clicking on it will reflect any possible changes, like if new tasks have started, etc...

More: Will open a window that shows more information on the currently selected node (note that a few list types won't use it). Once opened, you can leave this window open: it will simply adjust itself if you select a new list type. A few list types will add buttons to this window: they will be documented in the list type section of this manual. One button is common to all list types using this window: "Save". This button will allow you to save to a file the information listed in this window.

If you wish to have this window automaticaly opened at startup, you can set this in the preferences by enabling "Open More Window?".

The window can also be opened by double-clicking on a given node in the main list. Note that double-clicking on some columns already have an effect (like double-clicking on a task's pri column will prompt you to enter a new priority), so I recommend double-clicking on the name or on the address since these are sure to never execute a different action.

To close it, either click on its close gadget, or click again on the "More" button.

Menus

Project

About...	Information about SysInspector.
Iconify	Iconify the window.
Save to File...	Dumps the current list content to a file.
Quit	Quit SysInspector.

Settings

Prefs...	Configure SysInspector's settings.
Classact...	Configure ClassAct's GUI settings.

Tools

Flush memory...	Flush the memory from unused libraries/devices, invoke the low memory handlers, etc...
Selective Flush...	User-configurable memory flush.
Generate Report...	Goes through all the list types, and saves the output to a file.
System Settings...	Lets you enable or disable various system-related settings like CPU caches, moving VBR into Fast RAM, etc...
Intuition Settings...	Lets you change various settings related to Intuition, the GUI system of the Amiga:

Default PubScreen: Where windows asking for "default screen" will open.

Shanghai: All windows meant to open on Workbench will open on the default public screen.

Pop PubScreen: If a window opens on the default public screen, that screen will be brought to front.

Send to Explorer — This will send the address and the object type of the currently selected object in the main list to Explorer (launching it first if it isn't running). Explorer will disassemble the pointed object and let you browse through it. See the Explorer documentation for more details.

This won't work with the System, Interruptions, Commodities or Screenmode lists.

ARexx

Execute Script... — Will open a file requester, asking you to select an ARexx script to execute.

At this point, up to 10 entries can be added to this menu by the user through the preferences. This lets you select 10 ARexx script that will be directly accessible from there. See ARexx in the preferences.

1.8 Preferences

The prefs window consists of three pages, and the usual "Save" "Use" "Cancel" buttons found in most prefs editors. To flip through pages, click on the tabs located at the top of the window.

General

Deallocation: When you remove a task or a window, determines if SysInspector should try to free all resources it can, never do so, or ask you about it. Currently, SysInspector can:

\textdegree{} Close screens and windows opened by a killed task
\textdegree{} Free any message port the task had opened along with any queued messages
\textdegree{} Free any Gadtools gadgets attached to the window
\textdegree{} Free any Gadtools menus attached to the window

Iconified Position: Coordinates where you want SysInspector to put its icon while iconified. A value of -1 means "free position", where Workbench will select one.

Initial List: Select which list to display at startup.

Explorer Path: Select the path to the Explorer executable.

Options

Confirm Actions: Should SysInspector ask you to confirm before doing any action.

Strip Process Path: When displaying a command running from a CLI, should SysInspector only display the command's filename, without the whole path.

Snapshot Windows: When saving prefs, should the size and position of the "More" and the main window be saved.

Open More Window: Should the "More" window be opened at startup?

Format Hex Values: Should SysInspector format hex values ($1234 5678) or not ($12345678).

Use Button array: If you wish to replace the popup chooser with an array of 18 buttons to select the list type to display.

Menufy on Iconification: Instead of putting an AppIcon on the Workbench, will add an entry in the "Tools" menu.

Disable DOS Requesters: If SysInspector tries to access a non-available volume in the "Locks" and "Assigns" lists, don't display any requester asking for it.

Start Iconified: Should SysInspector be started in an iconified (or menufied) state.

Double Click Opens More: If you want SysInspector to open the More window on a double click in the list.

Button Array PopUp Help: When the pointer is stopped over a button of the button array, a small popup will appear, telling you what this button does.

You can also toggle this option by pressing on the "Help" key while in the main window.

Sort 1 and Sort 2

Configure how the items should be sorted for these list types:

Libs/Devs Resident Semaphores/Ports Assigns Tasks Mounts Screenmodes

Other list types will be listed in the same order as they are found on the system (some lists are already sorted by the system, such as the Memory nodes list.)

ARexx

This is where you can define a list of macros to be shown in the "Arexx" pulldown menu of the main window. You can use "Add" and "Remove" to add/remove entries, and you can double-click on an entry's column to edit it. Though you can enter as many entries as you wish, only the first 10 entries will be shown in the main window's "ARexx" menu.

The first column will contain the text shown in the pull-down menu, while the second column will contain the path and filename to the ARexx script.

Important note: The script path and filename must contain NO spaces at all. So instead of "Ram Disk:script.sirexx", use "RAM:script.siress", and blame CBM for putting a space in a system volume name.

Menus:

Project

Quit Prefs	Close the preference window.

Settings

Reset to Default	Restore the default settings.
Last Saved	Restore the last saved settings.
Restore	Restore the last used settings.

1.9 List Types

Here are the various lists SysInspector is able to display:

Assigns	Mounts
Commodities	Residents
Devices	Resources
Fonts	Screenmodes
Libraries	Screens/Windows
Locks	Semaphores
Memory	System
Memory Handlers	Tasks
Message Ports	Timer Requests

Resident DOS cmds

1.10 LIST: Assigns

This will show a list of DOS assigns currently on your system. Multiple assigns will be folded - click on the arrow pictogram to the left of the parent assign to reveal all the "child" (added) assigns.

Available Buttons:

Add: Will let you add an asssign to the system, as you would normaly do in a CLI with the C:Assign command.

Remove: Will remove the assign, freeing its associated lock at the same time. Unlike the AmigaDOS "Assign" command, this will let you remove a single assign from a multi-assign. Note that If you remove the parent assign of a multi-assign, all the child assigns will also be removed.

1.11 LIST: Commodities

This will show a list of the commodities currently loaded.

Available Buttons:

Remove: Will ask the commodity to quit, just like if you had done "Remove" in Exchange.

More: The "More" window will display four more buttons: "Enable", "Disable", "Show" and "Hide". These have the same functions as those the "Exchange" program, letting you enable/disable a commodity, and show/hide its interface (if it has one).

1.12 LIST: Devices

This will show a list of devices currently present in memory.

See "Libraries" for an explanation of the available options, they are the same.

1.13 LIST: Fonts

This will show a list of the fonts currently available in memory. Color fonts will have their name highlighted.

1.14 LIST: Screenmodes

This will show the screenmodes currently available in the display database.

1.15 LIST: Interrupts

This will give you a list of the interrupt handlers and servers currently installed on your system, sorted by interrupt types.

1.16 LIST: Libraries

This will show you a list of the libraries currently in memory. It also includes BOOPSI gadgets and datatype classes that have been loaded from disk.

Available Buttons:

Flush: Will try to expunge a library from memory. For this to succeed, the library's open count must be '0', meaning no application is currently keeping it open. Also, the library must allow itself to be expunged - some libraries don't.

Remove: Will forcibly remove a library from memory. What this really does is remove the library node from Exec's linked list of libraries, so note that no resource will be freed - the library will still be somewhere in memory.

NOTE: This is a RISKY operation! Only use this if you know what you're doing, like if you must remove a library you're working on that is stuck in memory, so you can load a new version.

In addition, double-clicking on the "OpenCnt" entry in the list will allow you to change it. Mostly of use when you are working on a library that got stuck in RAM because a program crashed without closing it, so you can't expunge it. Changing the open count will allow you to expunge it, meaning the Expunge() vector will be called on flushing unlike if you had simply did a "Remove" on it.

1.17 LIST: Locks

Will display a list of volumes. Clicking on the arrow pictogram to the left of a volume will expand the list, adding a list of all filelocks currently active on the volume.

Available Buttons:

```
   Remove: Will remove the lock, calling UnLock() on it.

           NOTES: This is a RISKY operation!  Only use this if the program
                  who installed the lock is crashed, suspended, or has
                  ended.
```

1.18 LIST: Memory nodes

```
This will display a list of the currently configured memory nodes.


Available Buttons:

   Add: This will let you add a new memory node, like non-Autoconfig
        memory.  You will be prompted to enter these informations:

           Node Name: The name to be given to that memory node.

           Base Address: The address at which the memory block starts.

           Size (KB): The size of the memory block to add, in KiloBytes.

           Priority: The priority of that memory node over the other ones.
                     Usualy, higher priority is given to fastest mem
                     (like 32-bits Fastmem), while the slowest mem such as
                     16-bits Fastmem or Chipmem will be given a lower one.

           Attributes:

              Local: Applies for memory that is located on the motherboard)

              Public: Memory that is publicly accessible by anyone.

              Kick: Memory that is available at the early stages of
                    system initialisation (unlikely to be the case if
                    you had to add it yourself thru SI ;)

           A few other attributes will be automaticaly set by SI:

              MEMF_24BITS will be set for memory blocks located within
              the first 24-bits of address space.

              Memory below $200000 gets MEMF_CHIP, or else it gets
              MEMF_FAST.



   Priority: This will let you change the priority of a memory node.
             Memory allocations will follow the node priorities, so
             when asking for MEMF_ANY, the memory node with the higher
             priority will be checked first for any free block.  So, it's
             logical to have a faster node have a higher priority over
             a slower one (like 32-bits RAM over 16-bits RAM).  Valid
             priorities ranges from -128 to 127.  Note that double-clicking
```

on the "Pri" column in the list will have the same effect.

1.19 LIST: Memory Handlers

This will show you the list of currently installed low memory handlers. These are usually called if a memory allocation failed. They will usually try to free some memory. Examples: a lowmem handler installed by a paint package might free some memory currently used for its undo buffer. The Ramlib lowmem handler installed by AmigaOS will try to expunge unused libraries from memory.

Available Buttons:

Priority: This will let you change the priority of a memory handler. Handlers are called in the order determined by their priority. Valid priorities ranges from -128 to 127. Note that double-clicking on the "Pri" column in the list will have the same effect.

Remove: This will remove a memory handler from the system.

NOTE: This is a RISKY operation! Only use this if you know what you're doing. It will merely remove it from the system (through exec.library RemMemHandler()), not freeing any resource.

1.20 LIST: Mounts

This will show a list of the mounted devices currently on your system.

More: When clicking on "Save", you will be asked if you want to generate a Mountlist. If you reply "No", then the window's content will simply be dumped to a file, just like for any other type.

NOTE: The FileSystem is NOT dumped in the Mountlist file. If someone knows how I can determine what filesystem is used by a mounted device, please tell me so.

1.21 LIST: Message Ports

This will show a list of the public message ports currently on your system.

Available Buttons:

Remove: Will close the message port. If you have "Confirm Actions?" set in your preferences, it will ask you if you want to reply any message(s) still queued to the port. Else, it will automaticaly do so.

NOTE: This is a RISKY operation! Only use it if you know what you're doing, like for removing a message port left behind by a crashed/terminated program.

1.22 LIST: Resources

This will show a list of resources currently present in memory.

Remove: Will forcibly remove a resource from memory.

NOTE: This is a RISKY operation! Only use this if you know what you're doing.

1.23 LIST: Resident

Will display a list of the resident modules, located in the ROM or added in RAM by the system during initialisation. Modules located in RAM will have their name highlighted.

1.24 LIST: Screens/Windows

This gives you a list of the currently opened screens. Each screen with at least one window opened will have a arrow pictogram to its left. Clicking on this will expand the list, inserting the list of opened windows just under the screen's node.

Available Buttons:

Remove: Will close the selected screen/window. If you try to close a screen that still has windows opened, it will also close these (asking you first to confirm the operation if you selected "Confirm Actions?" in the preferences). It will also ask you if you wish to deallocate any attached gadtools gadgets.

NOTE: This is a RISKY operation! Only use this if the task that opened the screen/window is gone, crashed, or suspended.

1.25 LIST: Semaphores

This will show a list of the public semaphores currently on your system.

Available Buttons:

Remove: Will remove the semaphore.

NOTE: This is a RISKY operation! Only use it if you know what you're doing, like for removing a semaphore left behind by a crashed/terminated program.

1.26 LIST: System

This will display various information about your system. Information is sorted in four categories. Clicking on the arrow pictogram to the left of a category will display its content.

Expansion Boards: Will display a list of expansion cards installed on your system, using boards.library to display their name and manufacturer.

NOTE: If SysInspector fails to recognize one of your expansion boards, Email the information to Torsten Bach, author of boards.library. You can contact him at lsi@berlin.snafu.de.

Hardware: CPU, FPU, CPU caches, gfx chipset, etc...
Note: the result for "gfx chipset" on the Draco is unknown. Tell me what happens (Enforcer hits, I suspect...).

Software: Kickstart, Workbench and Setpatch (if installed), graphic software version for CyberGFX/Picasso96/EGS (if installed), etc...

Vectors: Execbase vectors, Kicktags, etc...

CPU Traps: CPU Trap vectors, interrupt vectors, etc... Those are usualy pointing into ROM - vectors pointing to RAM (meaning they were modified) will be highlighted.

1.27 LIST: Tasks

This will show you all the tasks and processes being executed on your system. Note that some options are disabled if you select SysInspectorI's process, so to prevent you from crashing it by messing with it. (Ever been told that playing with yourself could make you deaf? Same thing with SysInspector - suspend it, and it will no longer hear your orders :)

Available Buttons:

Break: Send a break signal to the process/task. It is the same thing as using the "Break" AmigaDOS command.

You can send one of four different break signals:

```
          CTRL-C: This will usually terminate a process.
          CTRL-D: This will abort an AmigaDOS script.
          CTRL-E: Usually unused.
          CTRL-F: Sometime supported by some programs as a wakeup call,
                  to (re)open their GUI.


Remove: Will let you remove a task/process.  This is handy if a program
        just crashed on your system, and you wish to get rid of it.
        SysInspector will also let you free various resources.
        See "Task Dealloc" in the prefs.

        NOTE: This is a RISKY operation!  If you want to avoid the Guru,
              it is much safer to either just "Suspend" it (so it will
              no longer slow down your system eating CPU time), lower
              its task priority, or simply let it stay there.



Suspend: Remove the task/process from the active list, preventing Exec
         from scheduling it and giving it any CPU time.  This is handy
         if you wish to suspend the execution of a task/process, either
         because it's crashed or because it's slowing down your system.
         (I even used it once to pause a WB Tetris game I was playing :)
         You can achieve the same result by double-clicking on the
         node's "State" column.


Resume: Reactivate a suspended task/process.


Priority: Change the task priority.  Valid priorities ranges from -128
          to 127, but note that it's not recommended to use values
          outside of the -20 to +20 range.  You can achieve the same
          result by double-clicking on the node's "Pri" column.


Signal: Signal a task.  A new window will open, showing you
        the signal mask the task is waiting for, and 32
        checkboxes, each of them representing one of the 32
        possible signal bits.  Signal bits that the task isn't
        waiting for will show "--" as the bit number and will
        be disabled.  Just toggle the bits you want to signal,
        and click "Signal Task" to proceed.  You can also enter a value
        in the top right corner string gadget.  Hex values must start
        with a "$".
```

1.28 LIST: Timer Requests

```
When a program needs a time-triggered event to occur, it uses the
timer.device to setup a request.  When the asked time is elapsed,
timer.device will then signal the task.

This list will show you all the pending timer requests.
```

Available Buttons:

   Remove: Will abort a pending timer request, calling AbortIO() on it.
           Note that the waiting task will get signaled as if the event
           had naturally occured.

1.29 LIST: DOS Resident Commands

This is a list of DOS commands that are resident in memory (RAM or ROM).
This is the same list as you can manage using the "Resident" AmigaDOS
command from a CLI.

Available Buttons:

   Remove: Will remove a resident DOS command that had been added
           by the user.

   Disable: Will disable a resident DOS command that is present in ROM,
            allowing you to use a disk-based alternative instead of
            the ROM-based one.

1.30 ARexx Commands

   SysInspector has an ARexx port called "INSPECTOR.x", where "x" is
a value starting with 1, and increasing with each copies of SysInspector
being run at the same time. It is recommended, but not necessary that
scripts written for SysInspector use the .sirexx suffix.

Here's the list of the available commands:

  Preferences:

    GetPrefs   Get the current state of a preference element
    LoadPrefs   Load preferences
    OpenPrefs   Open the preferences window
    SavePrefs   Save preferences
    SetPrefs   Set a given preference element to a given state

  Main List:

    GetDisplay   Get the currently displayed list type
    GetNodeInfo   Obtain the content of the currently selected node
    GetStatus   Get the contain of the Status field in the main window
    Refresh   Refresh the list content
    SaveList   Save the currently displayed list to disk
    SaveReport   Save a complete report to a file
    SelectNode   Select a node in the currently displayed list
    SetDisplay   Set the displayed list to a given type
    SetStatus   Set the Status field in the main window.

```
  Node Functions:

     BreakNode          Send a break signal to the current task or process
     FlushNode          Attempt to expunge the current node from the system
     RemoveNode         Remove the current node from the system
     ResumeNode         Resume the current (suspended) task or process
     SendExplorer       Send the current node address to Explorer
     SetOpenCountNode   Set the open count of a library or device
     SetPriNode         Set the priority of the current node
     SignalNode         Signal the current task or process
     SuspendNode        Suspend the current task or process


  Miscellaneous:

     CloseMore       Close the More window
     FlushMem        Expunge memory
     Iconify         Iconify (or menufy) SysInspector
     OpenMore        Open the More window
     Quit            Terminate SysInspector
     SaveMore        Save the More window content to disk
     UnIconify       Reopen an iconified SysInspector
     Version         Return SysInspector's version string


   Arguments shown within [ ] are optional, while those shown within < >
are mandatory.  Most commands will return "ERROR" in RESULT if an error
occured.  An | indicates you possible arguments, i.e. for <a | b | c>, it
means you can specify either "a", "b", or "c".
```

1.31 ARexx - GetPrefs

```
COMMAND:

   GetPrefs - Return the value of the specified preference element.

USAGE:

   GetPrefs <Element>

DESCRIPTION:

   This will return the current value of the specified Element.  See
   SetPrefs for a list of valid elements.

   For Options, ON or OFF will be returned.

   For Sort Methods, "<value> - <type>" will be returned
      (example: 1 - By Name

   For others, a numeric value will be returned.
```

1.32 ARexx - LoadPrefs

COMMAND:

LoadPrefs - Load preferences from disk

USAGE:

LoadPrefs

DESCRIPTION:

This will load the preferences saved to disk in SysInspector.

1.33 ARexx - OpenPrefs

COMMAND:

OpenPrefs - Open the preference window

USAGE:

OpenPrefs

DESCRIPTION:

Will open the preference window, allowing the user to view or change settings, just as if he had opened it from the SysInspector pulldown menu.

1.34 ARexx - SavePrefs

COMMAND:

SavePrefs - Save preferences to disk

USAGE:

SavePrefs

DESCRIPTION:

This will save the current preferences to disk, just as if the user had selected "Save" in the Preferences Window.

1.35 ARexx - SetPrefs

```
COMMAND:

   SetPrefs - Set a given preferences element

USAGE:

   SetPrefs <Element> <Value>

DESCRIPTION:

   This will let you set the Element item to the value specified as Value.
   Here's a list of valid elements, a description of what they do, and
   their valid values:


     Sort Methods: (a value, 0 always meaning default order)

        LibDevSort      - Libraries and Devices
                              1 = By Name
                              2 = By Priority
                              3 = By Open count

        MountSort       - Mount entries
                              1 = By Name
                              2 = Volumes First

        ResidentSort    - Resident modules
                              1 = By Name
                              2 = By Priority
                              3 = By Type

        SemPortSort     - Semaphores and Message Ports
                              1 = By Name
                              2 = By Owner's name
                              3 = By Priority

        TaskSort        - Tasks and Processes
                              1 = By Name
                              2 = By Priority
                              3 = By State
                              4 = By Type

        AssignSort      - Assigns
                              1 = By Assign Name
                              2 = By Path
                              3 = By Type

        ScreenModeSort  - Screenmodes
                              1 = By Name
                              2 = By ModeID


     Options:  (Accepted values are "ON" or "OFF", to enable or disable the
               given option)
```

```
        Confirm            - Confirm Actions
        StripPath          - Strip process paths
        OpenMore           - Open the More Window at Startup
        Menufy             - Menufy when iconify
~       KillRequesters     - Kill DOS Requesters
        StartIconified     - Start Iconified
        FormatHexValues    - Format hex values
        DoubleClickForMore - DoubleClick opens More window
        UseButtonArray     - Use the button array
        PopupHelp          - Popup Help for the button array


     Miscellaneous: (Value in the specified range)

        WinX               - Main Window horizontal position (0 to 16000)
        WinY               - Main Window vertical position (0 to 16000)
        WinWidth           - Main Window width (200 to 16000)
        WinHeight          - Main Window height (100 to 16000)
        MoreWinX           - More Window horizontal position (0 to 16000)
        MoreWinY           - More Window vertical position (0 to 16000)
        MoreWinWidth       - More Window width  (50 to 16000)
        MoreWinHeight      - More Window height (25 to 16000)
        IconX              - Appicon horizontal position (0 to 2048)
        IconY              - Appicon vertical position (0 to 2048)

        InitialDisplay - Initial Displayed List, between 0 and 17.  Click
                         here for a list.


        Deallocate     - When resources should be deallocated, should SI:
                            0 = Free Any Resources
                            1 = Don't Free Them
                            2 = Ask

EXAMPLES:

   SetPrefs IconX 250 ; SetPrefs IconY 50
   SetPrefs Confirm ON
   SetPrefs TaskSort 1
```

1.36 ARexx - GetDisplay

```
COMMAND:

   GetDisplay - Returns the currently displayed list

USAGE:

   GetDisplay

DESCRIPTION:
```

Returns the number of the currently displayed list in RESULT.

1.37 ARexx - GetNodeInfo

COMMAND:

GetNodeInfo - Returns the content of the currently selected node.

USAGE:

GetNodeInfo [Column]

DESCRIPTION:

Returns the content of the specified column number (starting with 0) in RESULT. If no column is specified, then the whole line is returned.

If the specified column is invalid or no node is selected, ERROR is returned.

This won't work with lists that won't let you select a node (like the "System" list).

1.38 ARexx - GetStatus

COMMAND:

GetNodeInfo - Returns the content of the Status gadget

USAGE:

GetStatus

DESCRIPTION:

Returns the content of the main window's Status gadget in RESULT.

1.39 ARexx - Refresh

COMMAND:

Refresh - Refresh the currently displayed list

USAGE:

Refresh

DESCRIPTION:

Will do just like if you had clicked on the Refresh button on the main window, rescanning the system to rebuild the currently displayed list.

1.40 ARexx - SaveList

COMMAND:

SaveList - Save the content of the current list to a file.

USAGE:

SaveList <Filename>

DESCRIPTION:

Save the content of the currently displayed list to the specified file. If no Filename is specified, then a file requester will open, asking for one.

This is the same as doing "Save as ASCII..." in the main window's pulldown menu.

1.41 ARexx - SaveReport

COMMAND:

SaveReport - Save a complete report to a file.

USAGE:

SaveReport <Filename>

DESCRIPTION:

Save the content of every lists to the specified file. If no Filename is specified, then a file requester will open, asking for one.

This is the same as doing "Generate Report..." in the main window's pulldown menu.

1.42 ARexx - SetDisplay

COMMAND:

SetDisplay - Select the list type to display

USAGE:

```
  SetDisplay [List]


DESCRIPTION:

  Select what list to show in the main window.  The List argument must be
  one of the values accepted by the DISPLAY command line argument.
```

1.43 ARexx - SetStatus

```
COMMAND:

   GetNodeInfo - Set the content of the Status gadget


USAGE:

  SetStatus [Text]


DESCRIPTION:

  Set the main window's Status gadget to the specified Text.
```

1.44 ARexx - SelectNode

```
COMMAND:

   SelectNode - Select an entry in the displayed list.

USAGE:

   SelectNode <NUM=Number | ADDR=Address | Name>


DESCRIPTION:

   Selects the specified node in the displayed list.  A node can be
   specified in three different ways:

      1) By its position in the list  (SelectNode Number 0)

      2) By its address as shown in the first column.  Hexadecimal
         addresses can be spcified by using an "0x" prefix.
         (SelectNode Addr 0x100B40F3)

      3) By its name (SelectNode Name commodities.library)
         CASE-SENSITIVE.


   This won't work for lists that can't let you select a node, like the
```

"System" and "Interrupts" list. You can't select a Lock by its name either, because of the numerous nodes having the same name there.

1.45 ARexx - BreakNode

COMMAND:

BreakNode - Sends a break signal to the currently specified task

USAGE:

BreakNode [C | D | E | F]

DESCRIPTION:

Will send the specified break signal to the currently selected task or process, as if you had done a Ctrl-C, Ctrl-D, Ctrl-E or Ctrl-F in the process's input window. If no argument is specified, a requester will open, prompting you to select a signal to send.

Of course, this command is only valid in the "Task" list, and requires that a task or a process is selected in the list.

1.46 ARexx - FlushNode

COMMAND:

FlushNode - Attempt to expunge the currently selected node.

USAGE:

FlushNode

DESCRIPTION:

This will attempt to expunge the selected node from memory. The selected node must be a library or a device. Note that ERROR is only returned if FlushNode was attempted on a non-valid node (like a task). It won't report if the expunge actualy succeeded or not.

1.47 ARexx - RemoveNode

COMMAND:

RemoveNode - Remove the selected node from the system

```
USAGE:

   RemoveNode


DESCRIPTION:

   This will attempt to remove the currently selected node from the
   system, just as if you had clicked on the "Remove" button in the list.

   This command is invalid for a few lists (where the "Remove" button
   wouldn't be enabled anyway), like the "Fonts" or the "Memory" lists.
```

1.48 ARexx - ResumeNode

```
COMMAND:

   ResumeNode - Resume the currently selected task


USAGE:

   ResumeNode


DESCRIPTION:

   If the currently selected task or process was suspended, will resume it.
```

1.49 ARexx - SendExplorer

```
COMMAND:

   SendExplorer - Send the currently selected node address to Explorer.


USAGE:

   SendExplorer


DESCRIPTION:

   This will send the address of the currently selected node to  Explorer
   (launching it first if it wasn't already running).  Explorer will
   disassemble the pointed object and let you browse through it.

   This won't work with the System, Interruptions, Commodities or Screenmode
   lists.

   If Explorer couldn't be started, there was no node selected or the
   currently displayed list isn't supported, then 'ERROR' will be returned.
```

1.50 ARexx - SetOpenCountNode

```
COMMAND:

   SetOpenCountNode - Set the open count of a currently selected library or
                      device.


USAGE:

   SetOpenCountNode [OpenCount]


DESCRIPTION:

   Will change the open count of the currently selected library or device.
   The value must be between 0 and 65535.  If no value is specified, then
   a requester will open prompting you for one.
```

1.51 ARexx - SetPriNode

```
COMMAND:

   SetPriNode - Set the priority of the currently selected node


USAGE:

   SetPriNode [Priority]


DESCRIPTION:

   Will change the priority of the currently selected task, process,
   Memory or Memory Handler.  The priority must be a value between -128
   (lowest priority) and 127 (highest priority).  If no priority is
   specified, then a requester will open prompting you for one.
```

1.52 ARexx - SignalNode

```
COMMAND:

   SignalNode - Send a signal to the currently selected task or process


USAGE:

   SignalNode [SignalMask]


DESCRIPTION:
```

```
   Will signal the currently selected task or process with the specified
   SignalMask.  If no SignalMask is specified, then a window will open,
   asking you to select which signal bits to set.
```

1.53 ARexx - SuspendNode

```
COMMAND:

   SuspendNode - Suspend the currently selected task or process


USAGE:

   SuspendNode


DESCRIPTION:

   Will suspend the execution of the currently selected task or process,
   just like if you had clicked on "Suspend" in the main window.
```

1.54 ARexx - CloseMore

```
COMMAND:

   CloseMore - Close the More window.


USAGE:

   CloseMore


DESCRIPTION:

   If it's open, the More window will be closed.
```

1.55 ARexx - FlushMem

```
COMMAND:

   FlushMem - Flush memory.


USAGE:

   FlushMem [Devs] [Libs]
```

DESCRIPTION:

This will attempt to flush memory of no longer needed modules, just as if you had done an "Avail Flush" in a CLI. If Devs and/or Libs is specified, only those specified object(s) will be expunged from memory.

1.56 ARexx - Iconify

COMMAND:

Iconify - Iconify SysInspector

USAGE:

Iconify

DESCRIPTION:

This will close SI's main window, and add an AppIcon (or a new item in your Workbench's "Tools" menu if you have Menufy on Iconify enabled in your preferences).

1.57 ARexx - OpenMore

COMMAND:

OpenMore - Open the More window

USAGE:

OpenMore

DESCRIPTION:

This will open the More window, just as if you had clicked on the "More" button in the main window.

1.58 ARexx - Quit

COMMAND:

Quit - Terminates SysInspector

USAGE:

```
   Quit

DESCRIPTION:

   Quits SysInspector.
```

1.59 ARexx - SaveMore

```
COMMAND:

   SaveMore - Save the content of the More window to a file.

USAGE:

   SaveMore [Filename] [Mountlist]

DESCRIPTION:

   Save the content of the "More" window to the specified file.  A file
   requester will open if no Filename is specified.

   If you specify the Mountlist argument and the currently displayed item
   is a Mounted device, then the file will be in a valid Mountlist format.
```

1.60 ARexx UnIconify

```
COMMAND:

   UnIconify - Reopens SysInspector's window which was iconified.

USAGE:

   UnIconify

DESCRIPTION:

   If SysInspector was iconified or menufied, then it will remove the
   AppIcon/Menuitem and reopen the main window.
```

1.61 ARexx - Version

COMMAND:

Version - Return SysInspector's version string

USAGE:

Version

DESCRIPTION:

This will return SysInspector's version string in RESULT. It will have the following format:

SysInspector version.revision (dd.mm.yy)

Example:

SysInspector 1.4 (11.11.97)

1.62 Misc Notes

Usage tips:

- If you want to output a list's content to the current CLI, just save to a file named "*" (no path!). This will send the output to the current output window (stdout).

- When a program crashes or gets in an endless loop, the safest action to take is to lower its task priority to something like -20 so it won't slow down your system. Almost as safe is to Suspend it so it no longer use any CPU at all.

  Removing the task can be more risky, depending on a lot of factors. SI tries to make it as safe as possible, allowing you to chose what resources to free, but totally remove the task only if you can handle a potential crash. It might be safer to just suspend the task, and then close any window it opened through the "Screen/Windows" list, unless you also want SI to automaticaly take care of any msg port, allocated gadgets, etc...

- Remember: a system monitor can be a very powerful tool because it gives you access to private stuff of the operating system. When something is defined as being "risky" in the documentation, don't take it lightly.

- Some programs will refuse to run twice because they leave a public message port behind them (like an ARexx port). So if they crash, killing the old message port might allow you to run a new copy of it.

- If you use a large disk cache and want to flush unused libraries without also flushing your disk cache, the "Selective Flush..." item comes very handy.

- As I often do with my programs, I've put a secret message somewhere in it. Try to find the right hotkey to get it ;)

- If you are writing a library, and the test program crashes, leaving it opened so it can't be flushed, just remove/suspend the crashed program, and "Remove" the library in question. This will let you load a new (hopefuly fixed!) version in memory. You can also try changing the open count to 0, and then doing a "flush" on it.

- Most libraries/gadgets who are asked to flush themselves from memory while they are currently in use will set a "delayed expunge" flag. This means once the library/gadget opencount reaches 0, it will expunge itself from memory. This can be good to know if you try to flush a ClassAct gadget that's currently in use by SI - it will flushes itself once you quit SI, unless another application has opened it.

- When reporting bugs about a program, the author will usually like to get information on your system. Why not use SI to save the "System" information, and send that to them with your bug report? In some cases, the "Task" list and the "Libraries" list can also be of help to them.

Known Problems:

- On 640x200 screens there might not be enough room to display the complete 18 list types in the chooser popup. If the last(s) type(s) aren't shown, use the arrow on the chooser gadget to access those not accessible from the popup, use a taller screenmode, or use the button array.

- Mountlists generated through the More window's "Save" button for a mounted device will lack the "FileSystem" entry. Please tell me if you know how I could retrieve this information.

- The "Locks" list gets confused when two volumes have the same name.

- The "About" window animation will flicker quite a bit with older versions of button.gadget. It's not a bug, just something not yet fully supported in older versions of button.gadget.

  NOTE: ClassAct's button.gadget V42 has nothing to do with the older CBM one.

- Selecting a process written in E will show a bogus task usage. That's because E processes allocate their own stack space, but they don't update the Upper/Lower fields of the task structure. E programmers can use the fakestack module from Aminet (somewhere in dev/e) to avoid this problem - SI does.

- Some people asked me to add a memory/structure displayer. I don't intend to write one, simply because I feel I can't write something better than Explorer, from Jason Hulance :) I recommend grabbing it from Aminet (Explorer21j.lha is the current release, in dev/e), as well as the Amiga E main archive from that same directory (it uses the E modules to properly disassemble and display data structures). Makes a

```
   good companion to SI IMHO (I'll try writing some ARexx scripts to make
   it easier to interface both of them)

 - Trying to Flush glyph.image V41.4 or earlier will crash the system.
   This is fixed with glyph V41.5.


ToDO:

This is far from a complete list, but most notable are:

 - SCSI devices information

 - And a bunch of other details still cluttering my internal ToDo list
```

1.63 Explorer

```
[Explorer is a great system browser and debugger written by Jason Hulance.
 SysInspector can communicate with it to submit object pointers which will
 be displayed by Explorer.

 Explorer is included in this archive with Jason's permission.]



Explorer 2.2j
=============
Allows you to browse around memory as E objects (which are like C structs).


Basic Function
--------------
You enter an address and then select the object which is (supposed to be?)
at that address.  You can then see the value that each element of the
object takes, and follow pointers etc. by double-clicking on the element (a
single click changes the address to the element's address, which is an
offset from the address you specified for the object).

A double-click may therefore change the object being browsed, so there is a
way of returning to the original object via the "Back Up" button.  As you
double-click and select new objects your choices are recorded.  The current
depth of this path is shown as the 'Depth:' text.  The action of the "Back
Up" button is duplicated by the "Back Up" menu item on the "Selections"
menu.  There is also an option to clear the current path on this menu.

The list of objects may be created in several ways (these are found on the
"Project" menu):

 1) "Open" Opens a previously saved config file, containing any number of
           objects.  By default, when Explorer starts it tries to load
           objects from "explorer.config" in the current directory, and
           then "s:explorer.config" if that fails.  The supplied
           "explorer.config" file contains all the standard Amiga OS
           objects.  (This file was created by using "Load Modules" on the
           "Emodules:" directory and then saving the config using the next
```

option.)
2) "Save" Saves the current list of objects as an Explorer config file.
3) "Load Modules" Scan the selected module or all the modules in the selected directory and its sub-directories. The found objects will replace any current list of objects. This operation can take a long time, especially if you try to load all the modules in "Emodules:". You can interrupt it at any time by closing the status window.
4) "Merge Modules" Same as 3) but any objects found are added to the current list of objects rather than replacing them.
5) "Load Cache" Same as 3) but the current E module cache is scanned. This is most useful when Explorer is used to help debug your programs.

The address and object can also be specified via ARexx, making Explorer an extremely useful debugging tool as well as a system browser! In fact, EDBG v3.3a+ makes use of Explorer to analyse typed variables. (A module is supplied to show how easy it is to integrate the appropriate ARexx call into your program.)

Object Layout

The elements of the selected object are listed in order. If you choose to see the value of element it will be displayed in hex in parenthesis after the element, and if you choose to see the element address (an offset from the object address) it will be displayed after an '@'. (Use the "Selections" menu to decide which, if any, you want to see.)

Where to Start

Explorer starts off with the address of an internal hail and copyright string. The items on the "Presets" menu give some other, interesting starting points:

1) "Dos" Views the address stored in the 'dosbase' global variable as a 'doslibrary' object. The dos.library is one of the fundamental Amiga system libraries, and 'dosbase' is its corresponding library base object.
2) "Exec" Same as 1) but for the 'execbase' variable and object for the exec.library (the most fundamental Amiga system library).
3) "Graphics" Same as 1) but for the 'gfxbase' variable and object for the graphics.library.
4) "Intuition." Same as 1) but for the 'intuitionbase' variable and object for the intuition.library.
5) "This Process." Views the 'process' object for the running Explorer program!

For a typical Explorer session, you would probably start at one of these points and then double-click on elements and use the "Back Up" button to browse around. You never really need to set the address explicitly on the GUI (at worst you'd do it by ARexx), but the option is there if you really want to...

Simple Types

```
------------
The "CHAR", "INT" and "LONG" buttons view the contents of the address as if
it were a pointer to the corresponding type, and fills in the 'Hex:',
'Decimal:' and 'String:' texts with the value(s) being pointed at.

"BSTR" is similar to "CHAR" but it first converts the address from a BCPL
pointer and then skips over the length byte.  "BADDR" just converts the
address to a BCPL pointer.

"--" and "++" treat the address as if it were pointing to an array of the
type last requested (i.e., an object or simple type).  "--" moves the
address back to the previous element in the array, and "++" moves it on to
the next.  The size of the last requested type is shown as the 'Size:'
text.  (Note: "--" can be activated also by the "p" key and "++" by the "n"
key.)


Display Beeps!
--------------
The screen will flash (or you'll get a beep or whatever your Workbench is
set up to do) if you do one of the following:

 1) Select something when the address is NIL (zero).  A NIL address is not
    valid (although Explorer will behave as if it were).
 2) Select an element which is a NIL pointer or a LONG that is NIL.
 3) Select an element with object or PTR TO object type, where the object
    is not in the current list of objects.
 4) Try to backup past a depth of zero.


Command Line Options
--------------------
The command line template is:

    OBJECT,ADDR,ALL/S,CACHE/S,NONE/S,CONFIG/K,SCREEN/K,NOAUTOREPLY/S

 1) OBJECT Specify an initial object to be displayed (to be useful, the
           appropriate module must also be loaded using 'ALL' or 'CACHE').
 2) ADDR   Specify an initial address to be displayed (NIL is ignored!).
 3) ALL    Load all objects from the modules in Emodules: and its
           sub-directories (overrides 'CACHE' and 'CONFIG').
 4) CACHE  Load all objects from the modules in the current E module cache
           (overrides 'CONFIG').
 5) NONE   Start with no objects (overrides 'ALL', 'CACHE' and 'CONFIG').
 6) CONFIG Specify the config file to load.  If this load fails then
           Explorer will try "s:explorer.config".
 7) SCREEN Specify the public screen to open on.  Normally Explorer will
           open on the default public screen.
 8) NOAUTOREPLY Specifies that the "Auto Reply" option (described below)
           should start as off rather than on.

By default, Explorer will try to load the "explorer.config" in the current
directory (as if you'd used the CONFIG option -- so it will fall back to
trying "s:explorer.config").  The starting address will be that of an
internal hail and copyright string (which will be displayed as the
'String:' text).
```

ARexx

The "Reply" menu item on the "ARexx" menu manually replies to an ARexx message that may have been received. You can also let Explorer automatically reply to these messages by checking the "Auto Reply" option, but the manual method is best since you can be sure that the pointers remain valid while you browse (since the calling program is halted). Once you've replied to a message the calling program is free to scribble all over the memory at the address it just gave you...

A module 'sendexplorer' is supplied to simplify the debugging facilities of Explorer. It defines several functions:

1) PROC sendExplorer(addr,obj=NIL,repPort=NIL,message=NIL,quiet=FALSE)

  'addr' is the address you wish to use.
  'obj' is the name of the object you are interested in.
  'repPort' is a message port for replies. If you don't supply this then a temporary port is created for the duration of the function call. (The purpose of this parameter is to allow you to re-use the same port during your program if you call 'sendExplorer' a lot.)
  'message' is a short string to be displayed in the 'Message:' part of Explorer. Use this to communicate some information about the address and object (EDBG uses this to give the name, type and scope of the variable being explored).
  'quiet' is a boolean to say whether Explorer should complain if the object is not loaded.

The result is TRUE if the message was successfully delivered and did not cause an ARexx error, and FALSE otherwise.

An example call is:

  sendExplorer(packet, 'dospacket')

To browse using your own objects you'd separate them out into modules and load these modules in your program. Compiling your program would add these to the E module cache, which can then be used to set-up Explorer (using the "Load Cache" menu item). Alternatively, you could "Load" or "Merge" the appropriate module directly, or use a config file if the objects don't change very often.

2) PROC quitExplorer(repPort=NIL)

  'repPort' is as above.

This sends a 'QUIT' message to Explorer. Again, it returns TRUE if everything succeeded.

3) PROC isExplorerRunning()

Returns TRUE if Explorer is running. It works this out by testing for the presence of Explorer's ARexx port, so it may be fooled..

Altering Data

```
-------------
By popular demand, if you hold down the shift key when you click on an
element of an object or on the 'CHAR', 'INT' or 'LONG' buttons, then a
small window pops up which allows you to edit the appropriate value.  It's
not very safe to randomly edit data (especially system data), so use of
this feature is discouraged unless you really know what you're doing...


Example Program
---------------
To show how useful Explorer can be, there is a modified version of the
'empty' handler supplied.  This uses the above call to 'sendExplorer' to
communicate the value of the packet currently being processed.  Since the
handler is not a full process (it's just a task) there would normally be no
simple way of debugging it or examining the packets as they arrive (none of
the DOS functions, such as printing text, are available to simple tasks).
However, the handler can send messages to ARexx and thus to Explorer.

To try it out:

 1) Make sure ARexx is up (by running RexxMast, if necessary).
 2) Start up Explorer using,
      run explorer all
 3) Now copy the 'empty-handler' to L:,
      copy empty-handler L:
 4) Mount it using,
      mount empty: from empty.mountlist
 5) Use the handler,
      type empty:10
 6) Look at Explorer.  It will be displaying the first packet.  Browse
    around it, remembering that most elements will be BCPL strings.
 7) Select "Reply" from the ARexx menu (or press RAmiga-R) to reply to this
    message.
 8) Another packet arrives and the message is sent to Explorer.
 9) Repeat 6)-8) until the 'type' command finishes and you get the Shell
    prompt back.


EDBG
----
EDBG v3.3a+ communicates data about a variable if you hold down the shift
key and double-click on a variable that's within the current scope (EDBG
also adds it to the list of watched variables, if necessary).
```

1.64 LEGAL

```
    SysInspector is Copyright ©1997 by Eric Sauvageau.  You are allowed to
freely redistribute this program and its documentation.  Inclusion on PD
collections and coverdisks is allowed, but if you do so on a magazine
coverdisk, I'd appreciate that you send me a free issue of the magazine in
exchange.

   I decline any responsability for any problem encountered while using
SysInspector.  If it crashes your system, makes your dinner burn, or makes
Barney appear at night telling you he loves you, it's not my fault.  I'm
```

doing everything I can to ensure this product is as stable as possible, but bugs wouldn't exist if they were that easy to find. If you encounter such a critter, please report it to me so I can try to fix it in a future release.

SysInspector is released as Shareware. It ain't crippled in any way because I'm personally allergic to what I call "SabotagedWare", so I'm relying on YOU, the user. If you find it useful and worth paying for it, I'm asking you to send me 15$ US (20$ CAN). I can also accept a registered version of a program you wrote in exchange. The response from users will determine the future of this product. If no one feels it's worth registering, then I'll take it as a "it's not worth working on it".

Send money (only US or CAN dollars please) to:

Eric Sauvageau
5338 10th Avenue
Montreal, Qc
Canada
H1Y-2G6

Email:

merlin@thule.no

Please contact me by Email first before sending anything through regular mail, as I'll be moving during the first half of 1998.

The latest version (including public beta versions) will always be available from my homepage:

http://www.thule.no/~merlin/

1.65 Thanks

I want to thank these guys, who had something to do with SysInspector:

Philip Vedovatti	(beta-testing, artwork, Installer script)
Christopher Aldi	(beta-testing, ClassAct)
Petter Nilsen	(beta-testing, ClassAct)

```
  Christopher Gaul               (beta-testing)
  Andrew Robson                  (beta-testing)
  Tim Aston, Osma Ahvenlampi     (ClassAct)
  Torsten Bach                   (boards.library)
  Jason Hulance                  (Explorer)

And also thanks to the registered users for their support:

  Ben Monroe (USA)
  Magnus Holmgren (Sweden)
  Richard Körber (Germany)
  Ola Olsson (USA)
  Adam Hough (Canada)
  Michael D. Lewis (USA)
  Frank Bunton (Australia)
  Sylvain Bourcier (Canada)
  Andrew Gray (USA)
  Michael Baird (USA)
  Rémi Lenoir (USA)


Please consider registering.  This is the best way to ensure the future
development of this product.

Registered users will be notified through Email whenever a new public
version (beta or release) is available.
```

1.66 History

```
   1.4 - February 13th, 1998

       \textdegree{} NEW: Timer Requests and Resident DOS cmds added to the list  ↩
          types.

       \textdegree{} NEW: Added "Intuition Settings..." to the Tools menu.

       \textdegree{} NEW: "PUBSCREEN" argument/tooltype.

       \textdegree{} NEW: Added support for Jason Hulance's Explorer (included in
         this archive along with a pre-made datafile).

       \textdegree{} NEW: You can now open a new public screen from the "Screens/ ↩
          Windows"
         list.  Note that I have no intention of turning SysInspector into
         a full-featured public screen manager, so don't expect any fancy
         feature.

       \textdegree{} Fixed problems with RAM: DOS locks.  The "Scan RAM: locks"  ↩
          prefs
         option is now gone.

       \textdegree{}~New pictogram for Assigns.

       \textdegree{}~Message Ports priority is now editable.
```

\textdegree{} Iconifying when there was no icon available would use a garbled
icon (built-in icon had a wrng PlanePick value).

\textdegree{} boards.library 2.17 included (now recognizes 375 boards).

\textdegree{} Some changes to the listbrowser V42 support to match with the latest changes in the current beta.

\textdegree{} Animated "About" is back for button.gadget V41 (a forecoming update of the gadget will eliminate the flicker).

\textdegree{} Fixed potential Enforcer hits when saving prefs would fail.

\textdegree{}~Now uses 32-bits maths to calculate the largest block percentage
in the More window when displaying a memory node.

\textdegree{} Saving the content of the More window would display the wrong address for the displayed object if you were using formatted
hex values.

\textdegree{} Some changes to the Screenmode ModeID sort code, fixing some occasional problems with it.

\textdegree{}~After signaling a task, will wait for 3 secs instead of just one
before checking if it's gone.

\textdegree{}~Some changes to the "More" display for tasks and processes.

\textdegree{}~Note that the Warped FTP site is often behind with classes releases. I advise you check ftp://ftp.thule.no/pub/classact/
instead to obtain the latest ClassAct archive.

\textdegree{} I'll be probably be moving during spring '98, so please Email me
first before sending me any regular mail or registration fee.

1.3 - November 11th, 1997

\textdegree{} NEW: ARexx port with a complete command set.

\textdegree{} NEW: "ARexx" menu, with up to 10 user-defined entries. Check the new page in the preferences.

\textdegree{} NEW: "System Settings..." in the Tools menu.

\textdegree{} NEW: "Add" button for "Assigns" and "Memory" lists.

\textdegree{} NEW: Sortable "Screenmodes" list, and "Locks" are now sorted by names.

\textdegree{} NEW: Editable library/device OpenCount, by double-clicking on it.

```
    \textdegree{} When not using the button array, speedbar.gadget will no  ↩
       longer
      be opened, and pictograms won't be created, wasting less memory
      and screen pens in that mode.  Likewise, using the button array
      won't open chooser.gadget (unless actually required in another
      window).

    \textdegree{} You can now change SysInspector's own task priority from the
      Tasks list.

    \textdegree{} Closing a public screen will now try to remove any pending  ↩
       lock
      left after closing all open windows.

    \textdegree{} Various bugs removed from the Prefs window and the cleanup  ↩
       code.

    \textdegree{} No longer uses reqtools.library, now uses ClassAct's
      requester.class and getfile.gadget.  Also included an updated
      boards.library.

    \textdegree{} More information shown in the "System" list.

    \textdegree{} Various enhancements to the More window content.

    \textdegree{} Various enhancements/fixes to the GUI handling code.

    \textdegree{} CA prefs now launched async to SI.

    \textdegree{} Added support for the forecoming V42 of listbrowser.gadget.

    \textdegree{} Recompiled with EC 3.3a.

    \textdegree{} And probably quite a few more bug fixes here and there.



1.2 - June 7th, 1997

Only three registrations so far...  Humm...  You guys better show more
support, or I'll simply move on to other projects if nobody feels SI is
worth at least 15$ US...


    \textdegree{} NEW: Button array alternative for selecting the list type,
           with optional popup help.

    \textdegree{} NEW: Gadtools menu can also be freed when closing a
           window (or a task with an opened window).

    \textdegree{} NEW: Added CPU Traps to the "System" list.

    \textdegree{} NEW: Added a couple of new entries in the preferences.

    \textdegree{} Uses memory pools during list generation, making it a bit  ↩
       faster
      and fixing various random crashes happening on some systems.
```

```
\textdegree{} Improved the "Generate Report...".

\textdegree{}~More accurate sort method for "Mounts" sorted with volumes ↩
   first.

\textdegree{} "Flush Memory..." now reports how much memory was regained.

\textdegree{} Cleaned up the error handling code called if SI would fail to
  start.

\textdegree{} Left/Right cursor keys will change the list type.

\textdegree{} boards.library 2.12 included (now recognizes 355 boards).

\textdegree{} Dual Playfield Pri-2 would be shown as "DPF" instead of "DPF2 ↩
   " in
  the Screenmode list.

\textdegree{} Now also display screenmodes that don't have a defined name.

\textdegree{} 'Remove' and 'Flush' wouldn't work unless you had "Confirm
  Actions" enabled in the preferences.

\textdegree{} "System" can detect MBench under "Software".

\textdegree{} Some changes to various "More" lists.

\textdegree{} Removed the kludge from the System's Vector display now that
  LoadV43Module and FastExec can peacefuly coexist.  Make sure you
  have at least V2.7 of FastExec, and don't use the "FREEOLD" option.

\textdegree{} Removed the address padding kludge for hierarchical lists.  ↩
   Make
  sure you have listbrowser.gadget 41.313 or better.

\textdegree{} Background CLIs (launched with "Run") will be shown as
  CLI-BGND(<cli#>).

\textdegree{} Changed the small arrows in hierarchical lists with custom- ↩
   made
  ones (both larger and nicer).

\textdegree{}~A couple of other bugs were fixed.



1.1 - April 7th, 1997

    \textdegree{} NEW: TASKPRI argument in the commandline.

    \textdegree{} NEW: Mounts sort method can be configured.

    \textdegree{} NEW: "Generate Report..." in the Tools menu.

    \textdegree{} NEW: "Screenmodes" list type.  You will probably need to
           readjust your "Initial List" setting in your preferences and
```

```
            resave.

      \textdegree{} Had forgotten to put non-Newicon imageries for most icons.   ↩
         Oops :)

      \textdegree{} The More window won't unnecessarily refresh itself while  ↩
         disabled.

      \textdegree{} Added "Largest chunk" to the Memory's More window

      \textdegree{} Expansion boards now shown on two lines, and will also show
        if they're of ZorroII or ZorroIII type.

      \textdegree{} Assign list now supports paths upto 150 chars long (still  ↩
         only the
        first 50 chars displayed).  Should fix the "Fault 120" errors
        happening too often to those with long path names.

      \textdegree{} SI will stop playing shy and will display its own info in the
        More window when its task is selected.


  1.0 - March 23th, 1997

      \textdegree{} First Aminet release.


  ß0.050 - December 29th, 1996

      \textdegree{} First public beta release.




"I don't think I'm amazing
 In fact, I'm quite insane
 To live inside my bucket
 With all my plastic chains."

     - Ozzy Osbourne.
```

1.67 Other Programs I wrote

```
  Other programs that I wrote:

       \textdegree{} CAMP 1.2 - ClassAct Mp3 Player is a small GUI front-end
                    for Stéphane Tavenard's mpega, an mp3 audio decoder.

       \textdegree{} DevsMan 1.4 - Devs: directory manager, allow easy handling
                      for your DOSDrivers, Datatypes, etc...
```

```
\textdegree{} FileScroller 3.40 - File lister for TransAmiga BBS (3.50 and ↩
    up
                    for Excelsior! BBS.)

\textdegree{} LowFrag 1.3 - Small system patch that helps reducing memory
               fragmentation.

\textdegree{} MFormat 1.8a - Replacement for CBM's "Format" command.  Has  ↩
   a
                complete GUI, configurable device filter, can
                install a bootable bootblock, etc...

\textdegree{} NewIcons V4.1 - Replaces the icon system with a new one that
                 allows icons to have palette information stored
                 in them, and to be properly remapped no matter
                 what's your WB palette, and still more.

\textdegree{} TDPrefs 1.0 - Preferences editor for trackdisk.device, can
               adjust the step rate, disable the drive click,
               etc...

\textdegree{} XPKatana 1.2  - GUI/ARexx interface for the XPK packing  ↩
   system,
                let's you (un)pack files using standard XPK
                libraries, can also unpack files packed with alien
                formats through xfdmaster.library.
```