Scout

System Monitor
Scout 37.227 (Release 3.0)
Edition 3.0
May 2002

by Thore Böckelmann

1 Introduction

What is Scout?

`Scout` is a tool that allows you to monitor your computer system. It displays many different things — like tasks, ports, assigns, expansion boards, resident commands, interrupts, etc. — and you can perform some certain actions on them.

For example you can freeze tasks, close windows and screens, release semaphores or remove locks, ports and interrupts.

Through `AmiTCP` it's also possible to use `Scout` as an TCP/IP service.

Since version 2.0 of `Scout` you can use nearly all implemented functions through shell parameters. Therefore it's not necessary to install `MUI` for using `Scout`, but you will need `MUI`, if you want to use `Scout` with its graphical user interface.

2 Legalities

Copyright

Please read the following parts carefully. You accept the following terms by starting the software, even for a test drive only.

COPYRIGHT

Scout 37.227 (Release 3.0) - Copyright © 1994-2002 by Andreas Gelhausen, Richard Körber and Thore Böckelmann, all rights reserved.

You only have the right to use the software, but no rights on the software itself. Disassembling, resourcing and all other ways of reverse engineering is forbidden.

FREEWARE

Scout is FreeWare. You are allowed to use the packet without paying a fee or similar to the authors. Of course we would appreciate a small donor! ;-)

COPYING

You can copy the packet as long as it remains entire and unchanged.

You are allowed to compress the packet using a customary compression software (as lha, lzh, lzx, dms). You must not compress single files of the packet (e.g. PowerPacker or Imploder).

DISTRIBUTION

You must not exceed an usual price on the market for your working and material. This means a maximum of 5 DM (or the equivalent amount in other currencies, including all taxes) for disks and 35 DM for CD-ROMs containing a PD software collection.

In any case, you need a written permission from us if you want to include Scout on a cover disk or in connection with a commercial product.

We explicitly permit the distribution via AmiNet, Meeting Pearls and Fred Fish.

LIABILITY

You are using the program as it is, with all flaws, and on your own risk! We grant no warranty for the software meeting a special purpose. This software may cause financial damage or harm people.

LIMITATIONS

You are not allowed to use this software and its results

- for fascism or military purposes
- if you do not agree to the copyright note

In this case you must delete the software and all related and generated files immediately!

CONTENTS OF THE PACKET

The Scout packet is *only* entire with these files:

```
Scout/arexx/activatetask.scout
Scout/arexx/arexx.readme
Scout/arexx/arexx.readme.info
Scout/arexx/breaktask.scout
Scout/arexx/freezetask.scout
Scout/arexx/poptofront.scout
Scout/arexx/settaskpri.scout
Scout/arexx/startup.scout
Scout/arexx.info
Scout/help/deutsch/scout-39.guide
Scout/help/deutsch/scout-39.guide.info
Scout/help/deutsch/scout.doc
Scout/help/deutsch/scout.doc.info
Scout/help/deutsch/scout.dvi
Scout/help/deutsch/scout.guide
Scout/help/deutsch/scout.guide.info
Scout/help/deutsch.info
Scout/help/english/scout-39.guide
Scout/help/english/scout-39.guide.info
Scout/help/english/scout.doc
Scout/help/english/scout.doc.info
Scout/help/english/scout.dvi
Scout/help/english/scout.guide
Scout/help/english/scout.guide.info
Scout/help/english.info
Scout/help.info
Scout/icons/icons.readme
Scout/icons/icons.readme.info
Scout/icons/Scout.info
Scout/icons/ScoutDock
Scout/icons/ScoutDock.info
Scout/icons.info
Scout/libs/identify.library
Scout/libs/identify.readme
Scout/libs/identify.readme.info
Scout/libs.info
Scout/Product-Info
```

```
Scout/Scout
Scout/Scout.info
Scout/scout.history
Scout/Scout.history.info
Scout/Scout.readme
Scout/Scout.readme.info
Scout.info
```

TRADEMARKS

3 Before Starting

3.1 System Requirements

`Scout` only requires AmigaOS version 2.04. The `identify.library` V8 is not required, but I strongly suggest to install it to give Scout its full power. See also Section 3.3 [Identify], page 7.

If you want to use `Scout's` graphical user interface, you also have to install `MUI` version 2.1 or higher. See also Section 3.2 [MUI], page 7.

The TCP/IP features of `Scout` are only available, if you have installed the version 4.0 of `AmiTCP`. See also Section 3.4 [AmiTCP], page 7.

3.2 MUI - MagicUserInterface

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called '`muiXXusr.lha`' (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send DM 30.- or US$ 20.- to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

3.3 Identify

`Identify` is a Shared Library that decodes expansion IDs, guru codes and library functions, and identifies your system.

`Identify` is FreeWare. You can find a complete package in the AmiNet ('`util/libs/Identify.lha`') or on the author's home page: `http://www.is-koeln.de/einwohner/shred/`. To get in contact, write to `shred@chessy.aworld.de`.

3.4 AmiTCP

`AmiTCP` is a TCP/IP protocol stack for the Amiga. The demo version 4.0 (or higher) should be available in greater public domain collections or on the AmiNet. Ask your preferred Amiga dealer. =:^)

Installing Scout

You only have to copy the program `scout` to your favourite directory, and '`identify.library`' to '`libs:`'. Then you can start it.

4 How to use Scout

This chapter describes the usage of `Scout` through its graphical user interface. This graphical user interface is based on the `Magic User Interface` (`MUI`) and `MUI` have to be installed in your system, if you want to use `Scout` trough windows and so on.

If you don't like `MUI`, you should see Section 4.30 [Scout without MUI], page 33.

If you start the program you will see the main window which includes many gadgets. Each of these gadgets represents a certain kind of system structure.

You can choose between:

Allocations, Assigns, BoopsiClasses, Commodities, Devices, Expansions, Fonts, InputHandlers, Interrupts, Libraries, Locks, LowMemory, Memory, Mounted Devices, Ports, Resident Commands, Residents, Resources, ScreenMode, Semaphores, System, Tasks, Timer, Vectors, Windows, Patches, Catalogs and AudioModes.

Click one of these gadgets and another window will be opened with a list of the structure type that is indicated on the pressed gadget.

> **Example:** Press the task gadget and you will get a window with the list of tasks and processes.

You can also select these functions by pressing the underlined key you see on each gadget or by using the right mousebutton menu.

If you wish to handle/remove a given structure, you should know what you do.

> **Warning:** Wrong handling of the showed structures can crash your system. At the worst you will lose your data.

Please note: You should not be surprised, if you don't find a certain detail information in this manual, because it's to much work to explain each element of all the structures you could see in this program.

Many books are written about these things and if you want to have more information about them, you should have a look in the specialized literature.

4.1 Allocations

This window informs about who allocated what hardware resource.

CIA

The Amiga owns two CIA to control its hardware, the keyboard and the printer interfaces. Additionally, it contains a couple of timer. This

window shows which parts of the CIA are not yet allocated, or which program allocated the resource.

'`Timer A`'
'`Timer B`' This are two 16bit timer. They can also be coupled to a 32bit timer.

'`Alarm`' This resource informs if a third timer reached an alarm value.

'`Serial`' This is a simple serial interface. Usually, the CIA-A one's is used for communicating with the keyboard. The CIA-B serial interface will not be allocated in most cases.

'`Flag`' This is a special control line. It is connected from the CIA-A to the Index line of the floppy disk drives.

Note: newer DraCos do not contain the CIA chips. Thus, these hardware resources will be emulated or are even not offered.

Ports

This are the resources for the internal parallel and serial interfaces.

'`Serial Port`'
This are the plain data transfer registers (transmitting and receiving).

'`Serial Control`'
This are the serial control lines, as Carrier Detect.

'`Parallel Port`'
This are the data lines of the parallel port.

'`Parallel Control`'
This are the control lines of the parallel port, as Busy or Paper Out.

Actions

'`Update`' The window is updates each time you press this button.

'`Print`' The window's contents are printed or saved to a file of your choice.

'`Exit`' The window will be closed.

4.2 Assigns

This type of structure assigns a logical name to a directory.

If you assign the directory '`dh0:data/documents`' the logical name '`texts:`', you will also be able to choose a file *filename* in that directory with the path 'texts:*filename*'.

Column items

'`Address`' Address of the assign structure.

'`Name`' Logical name of a directory

'`Path`' Here you will find the path of the directory.

Actions

'`Update`' Selecting this gadget updates the list of assigns.

'`Print`' This function allows you to send the list of '`Assigns`' to printer or a selected file.

'`Remove`' The selected assign will be removed with this function.

'`Exit`' The '`Assigns`' window will be closed.

4.3 BoopsiClasses

BOOPSI classes are object oriented classes. The classes in this list are all classes that are publically available from intuition.

Column items

'`Address`' This is the address of the `IClass` structure, which contains all data of this class.

'`Objects`' Shows the current amount of objects constructed by this class.

'`Subclasses`'
Shows the current amount of sub classes (public and private) which are derived from this class.

'`Superclass`'
A pointer to the `IClass` structure of the parent class.

'`Dispatcher`'
A pointer to the dispatcher code, which realizes all methods of the class.

'`Name`' The name of the class.

Actions

'`Update`' The list will be updated.

'`Print`' This function allows you to send the list to printer or a selected file.

'Remove' Removes the selected class from the system. This is only possible if neither objects nor sub classes exist from this class.

'More' Opens a window with further information.

'Exit' Closes the window.

4.4 Commodities

Commodities are small utilities. Most of them react on the input stream, or manipulate it.

You can find some commodity examples in the 'Tools' drawer of your Workbench.

Column items

'Address' Points to the CxObj structure of the commodity, containing all data about it.

'ln_Type' Structure type. Usually, it will be Broker.

'ln_Pri' Priority of the commodity broker.

'Flags' Flags describing the broker.

'Port' All messages of the broker are sent to this MessagePort.

'Name' Name of the commodity.

Actions

'Appear'
'Disappear'
Let the selected commodity's GUI pop up or disappear. Some commodities do not have a GUI.

'Enable'
'Disable' The commodity will be enabled or disabled.

'Kill' Let the selected commodity remove itself in a clean way.

'ListChg'
'Unique' The commodity is notified that the list has been changed or that another commodity with the same name was about to be added. This is only useful for programmers to test out their commmodities.

'Update' The list will be updated.

'Print' This function allows you to send the list to printer or a selected file.

'`Priority`'
: This function allows you to change the priority of a commodity.

'`Remove`'
: Removes the selected commodity from the system. Please try a frienly remove before, using '`Kill`'. Maybe the commodity removes itself voluntarily. =;^)

'`More`'
: Opens a window with further information.

'`Exit`'
: Closes the window.

4.5 Devices

A device is — like a library (see Section 4.10 [Libraries], page 17) — a collection of functions/procedures, which have to do certain jobs.

E.g. the '`trackdisk.device`' includes functions for the floppy disk handling.

Column items

'`Address`'
: Address of the device structure

'`ln_Name`'
: Name of a device

'`ln_Pri`'
: Priority of a device

'`OpenC`'
: This element shows how often the device was opened.

'`RPC`'
: '`RPC`' means '`RAM Pointer Count`' and shows how many jump addresses of the device point into RAM. In this way many programs — like the `setpatch` command from Commodore — patch the system.

  Many viruses patch the system in this way too, but don't panic now. If you check your system in regular intervals with a current virus killer, it should be out of danger.

  If the whole program code of the device is located in RAM, you will find a dash (minus sign) here, because in this case it's unimportant how many jump addresses point into RAM.

'`ln_Type`'
: Type of this structure (usually '`device`')

Actions

'`Update`'
: If you select this gadget, the list of devices will be updated.

'`Print`'
: This function allows you to send the list of '`Devices`' to printer or a selected file.

'Remove' The selected device will be removed with this function provided that no program uses this device anymore and the 'OpenC' is zero.

'Priority'
Herewith the priority of the device can be changed. A little window will be opened, that asks you for a new priority. Through the new priority it can happen that the device gets a new place in the device list.

'More' Another window will be opened and you will see more informations about the selected device.

You will have the same effect, if you doubleclick an element of the device list.

'Functions'
All device function offsets and addresses are shown up in a subwindow. If an appropriate '.fd' file exists and an 'FD:' assign points to its directory, then you will also see the function names.

'Exit' The 'Devices' window will be closed.

4.6 Expansions

IMPORTANT: All Scout releases before 2.10 are not compatible to this release! So if you want to read this list using TCP/IP, make sure that the remote system runs the latest version as well!

This window shows a list about all your expansion boards (graphic boards, memory expansions and so on) too.

Column items

'Address' The address of the expansion structure.

'BoardAddr'
Usually you will find the ROM of the card here. If this address points into RAM, the card is a memory expansion.

'Type' Information about the board. See the More window for further information.

'Manufacturer'
Name of the board manufacturer.

'Product' Name and class of the product.

Additional information

If you select one item from the list, the text field below shows up some additional information:

'`Size`' If the entry belongs to a memory expansion, the size of the memory is displayed here. Otherwise it's the ROM size of the card.

'`Flags`' See the More window.

'`ID`' ManufacturerID, assigned by Commodore, followed by the Productnumber, assigned by the manufacturer of the board.

'`SN`' Serialnumber of the card (usually unused)

Actions

'`Print`' This function allows you to send the list of '`Expansions`' to printer or a selected file.

'`More`' Now a window will be opened, that includes more informations about the selected expansion board.

Doubleclick an element of the '`Expansions`' list and you will have the same effect.

'`Exit`' The '`Expansions`' window will be closed.

Unknown expansion boards

If you select an expansion board by selecting its list item, you will get the name of the manufacturer and the card in the textfield you find below the list, provided that the installed version of '`identify.library`' knows about these data.

If no information is available in this textfield or the given information is wrong, you should send me the following data, please.

1. `ManufacturerID (Manufacturer)`
2. `ProductID (Product)`
3. `Name of the company`
4. `Name of your expansion card`
5. `Function of your card`

If you send me these data, the next version of '`Identify`' will include your expansion boards. Please be as precise you can.

4.7 Fonts

This function will show you all fonts existing in your system.

Column items

'YSize' Vertical size of the font

'Count' Here you can see how many programs use the font.

'Type' 'ROMFONT' means the font is located in ROM and 'DISKFONT' means the font was loaded from disk/harddisk.

'Name' Name of the font

Actions

'Update' The list of fonts will be updated.

'Print' This function allows you to send the list of 'Fonts' to printer or a selected file.

'Close' The font will be closed by using this function.

'Remove' It is possible to remove a font from system, provided that no program uses it and it's no 'ROMFONT'.

'Exit' The 'Fonts' window disappears.

4.8 InputHandlers

Input handlers take care of all user input arriving in system (pressed keys, mouseclicks, inserted disks, etc.). They stand one behind the other like on a production line and analyze the user input. The input handler with the highest priority gets the 'events' first and if it doesn't know how to react on these 'events', the second input handler gets them, and so on.

Usually the system input handler has a priority of 50. Every input handler, that wants to get the user input before the system, must have a higher priority.

Column items

'ln_Name' Name of the input handler

'ln_Pri' Its priority

'is_Data' This address points to some data needed by the input handler.

'is_Code' The program code starts here. If the code is located in RAM, the address is of different color. Otherwise you can find the code in ROM. Some viruses install an input handler in system. In this case the 'is_Code' address points into RAM, but many other programs uses input handlers, too. Don't panic!

Actions

'`Update`' The list of input handlers will be updated when you select this gadget.

'`Print`' This function allows you to send the list of '`InputHandlers`' to printer or a selected file.

'`Remove`' Removes an input handler from system.

'`Priority`'
Changes the priority of an input handler.

'`Exit`' The window will be closed.

4.9 Interrupts

Interrupts are important events the computer system has to react on. It exists a list of interrupt routines for each interrupt type. If a certain interrupt occurs, all these interrupt routines will be called. During their execution the running program will be interrupted.

Column items

'`ln_Name`' Name of the interrupt

'`ln_Pri`' Its priority

'`is_Data`' At this address you find the data of the interrupt.

'`is_Code`' Address of the interrupt code. If this address points into RAM, it's of a different color.

'`NUM`' This number represents the type of event the interrupt routine is called on. The '`IntName`' you find in the interrupt detail window gives you a little bit more information about it.
Example: Number 5 means that the interrupt is called at every vertical blank interval.

Actions

'`Update`' The list of interrupts will be updated.

'`Print`' This function allows you to send the list of '`Interrupts`' to printer or a selected file.

'`Remove`' If the interrupt is a server you can remove it from system. An interrupt handler can't be removed by `Scout`.
If you call '`avail flush`' and the audio.device isn't used, the interrupt handlers of the audio.device will be removed.

'`More`' Now a window will be opened that includes more details of the interrupt.

'`Exit`' Selecting this gadget will close the '`Interrupts`' window.

4.10 Libraries

A library is a collection of functions/procedures, which have to do certain jobs. E.g. the 'graphics.library' includes routines for graphical display.

Column items

'`Address`' Adress of the library structure

'`ln_Name`' Name of a library

'`ln_Pri`' Priority of a library

'`OpenC`' Here you see, how often the library was opened.

'`RPC`' '`RPC`' means '`RAM Pointer Count`' and shows how many jump addresses of the library point into RAM. In this way many programs — like the `setpatch` command from Commodore — patch the system.

Many viruses patch the system in this way too, but don't panic now. If you check your system in regular intervals with a current virus killer, it should be out of danger.

If the whole program code of the library is located in RAM, you will find a dash (minus sign) here, because in this case it's unimportant how many jump addresses point into RAM.

'`ln_Type`' Type of this structure (usually '`library`')

Actions

'`Priority`'

Herewith the priority of the library can be changed. A little window will be opened, that asks you for a new priority. Through the new priority it can happen that the library gets a new place in the list of libraries.

'`Close`' A library must be closed by all programs, if you want to remove it from system. In this case the '`OpenC`' is zero.

If you select this function, you will be asked, how often you want to close it. You can choose between '`Once`' and '`All`'.

Select '`All`' and the library will so often be closed till the '`OpenC`' is zero.

'Remove' The selected library will be removed with this function provided that no program uses this library anymore and the 'OpenC' is zero.

Some libraries can't be removed from system without a reset. So you shouldn't wonder about it, if this happens.

'Functions'
All library function offsets and addresses are shown up in a subwindow. If an appropriate '.fd' file exists and an 'FD:' assign points to its directory, then you will also see the function names.

'Update' The list of libraries will be updated.

'Print' This function allows you to send the list of 'Libraries' to printer or a selected file.

'More' A window will be opened that includes more details of the library.

'Exit' Selecting this gadget will close the 'library' window.

4.11 Locks

A lock structure shows you, that a program reads from or perhaps write into a file or a directory. With this type of structure the system prevents, that a file will be deleted while another program gets some data from it.

Column items

'Access' Here you can see the type of access. This could be 'READ', 'WRITE' or 'OWN'. 'OWN' stands for a lock Scout created to get the elements of this list.

'Path' Path of the file or directory

Actions

'Update' The list of 'Locks' will be updated.

'Print' This function allows you to send the list of 'Locks' to printer or a selected file.

'Remove' A lock will be removed through dos.library's 'UnLock()' function.

'Pattern' If you give Scout a pattern, only the locks with a matching path will be shown.

'Exit' The 'Locks' window will be closed.

4.12 LowMemory

This list contains all low memory handler known to the system.

These handlers are called in their sequence if a memory allocation is about to fail due to missing resources. The handlers try to free as much unused memory space as possible.

'ramlib' is such a low memory handler removing unused libraries and devices from the system's memory. It is always present.

Note: Low memory handlers are only available since AmigaOS 3.0. On older systems this list will always be empty.

Column items

'`Address`' Address of the low memory handler structure.

'`ln_Name`' Name of the handler.

'`ln_Type`' Type of the handler.

'`ln_Pri`' Priority of the handler. All handlers are called in their priority sequence.

'`is_Data`' A pointer to some handler's private data.

'`is_Code`' A pointer to the low memory handler code.

Actions

'`Update`' The list will be actualized.

'`Print`' This function allows you to send the list to printer or a selected file.

'`Remove`' The low memory handler will be removed from system.

'`Priority`'
Changes the priority of the selected handler.

'`Exit`' The window will be closed.

4.13 Memory

In this list you will find the segments of your memory. At least you will find an entry for your chip memory.

Column items

'`ln_Name`' Name of the memory segment (e.g. '`chip memory`')

'`ln_Pri`' Priority of memory

'`mh_Lower`'
First address of memory

'`mh_Upper`'
Last address of memory

Actions

'`Print`' This function allows you to send the list of the memory segments to printer or a selected file.

'`Priority`'
This function allows you to change the priority of a memory segment. The memory segment with the highest priority will be preferred from system, provided that no certain type of memory is demanded.

'`More`' Another window will be opened. This window includes more information about the memory segment.

'`Exit`' The window will be closed.

4.14 Mounted Devices

In this list you will find all your devices like disk drives, printer devices, etc.

Column items

'`Name`' Name of the device

'`Unit`' Unit number

'`Heads`' Number of heads

'`Cyl`' Number of cylinders

'`State`' The state shows you for example, if a disk is in drive.

'`DiskType`'
Type of a disk (e.g. OFS (OldFileSystem), FFS (FastFileSystem), ...)

'`Handler or Device`'
The handler or the device you find here has to manage the stream of data from and to the device.

Actions

'`Update`' The list will be updated.

'`Print`' This function allows you to send the list of '`Mounted Devs`' to printer or a selected file.

'`Exit`' The window will be closed.

4.15 Ports

Programs are able to communicate together through ports.

Column items

'`Address`' Here you will find the port structure.

'`ln_Name`' Name of port

'`ln_Pri`' Priority of port

'`mp_SigTask`'
The task is communicating through the port.

Actions

'`Update`' The ports list will be updated.

'`Print`' This function allows you to send the list of '`Ports`' to printer or a selected file.

'`Remove`' The port will be removed.

'`Priority`'
Herewith the port priority can be changed.

'`Exit`' The '`Ports`' window will be closed.

4.16 Resident Commands

This list includes all resident commands. That means all commands you find in ROM and the commands you made 'resident' through the `resident` command.
Positions and sizes of their hunks you will find here, too.

Column items

'`Name`' Name of the command

'`UseCount`'
Here you can see, how often a command was being executed at the time the list was build.

'`Lower`' First address of hunk in memory

'`Upper`' Last address of hunk in memory

'`Size`' Size of hunk (upper - lower - 8 bytes overhead)

Actions

'`Update`' The list of '`Resident Commands`' will be updated.

'`Print`' This function allows you to send the list of '`Resident Commands`' to printer or a selected file.

'`Remove`' The selected command will be removed with this function provided that no program uses this command anymore and the '`UseCount`' is zero.

'`Exit`' The window disappears.

4.17 Residents

Resident modules are reset-protected segments (code and data). In the list of '`Residents`' you usually find libraries, devices and resources. A programmer has the possibility to make his own programs reset-protected. He has to initialize a resident structure for it and then he can link the program through the kick-vectors (see Section 4.24 [Vectors], page 29) to the list of the resident modules. The residents you linked to system are usually located in RAM and are of a different color.

If you find a resident module that points into RAM and you don't know which program has created it, you should start your favourite virus detector and let it check your memory. Many viruses prefer this way to travel around.

Column items

'`Address`' At this address the resident module is located.

'`ln_Name`' Name of the resident module

'`rt_Pri`' Priority

'`rt_IdString`'
Identity string of the resident module.

Actions

'`Update`' The list of '`Residents`' will be updated.

'`Print`' This function allows you to send the list of '`Residents`' to printer or a selected file.

'`More`' Selecting this gadget opens a new window with more information about the selected resident module.

'`Exit`' The '`Residents`' window will be closed.

4.18 Resources

Usually a resource is — like a library (see Section 4.10 [Libraries], page 17) — a collection of functions/procedures, which have to do certain jobs.

E.g. the 'filesystem.resource' includes functions for the filesystem handling.

Column items

'`Address`' Address of the resource structure

'`ln_Name`' Name of a resource

'`ln_Pri`' Priority of a resource

'`OpenC`' This element shows how often the resource was opened.

'`RPC`' '`RPC`' means '`RAM Pointer Count`' and shows how many jump addresses of the resource point into RAM. In this way many programs — like the `setpatch` command from Commodore — patch the system.

Many viruses patch the system in this way too, but don't panic now. If you check your system in regular intervals with a current virus killer, it should be out of danger.

If the whole program code of the resource is located in RAM, you will find a dash (minus sign) here, because in this case it's unimportant how many jump addresses point into RAM.

'`ln_Type`' Type of this structure (usually '`resource`')

Actions

'`Update`' The list of '`Resources`' will be updated.

'`Print`' This function allows you to send the list of '`Resources`' to printer or a selected file.

'Remove' The selected resource will be removed with this function, provided that no program uses it anymore and the 'OpenC' is zero.

'Priority'
Herewith the priority of the resource can be changed. A small window will be opened, that asks you for a new priority. Through the new priority it can happen that the resource gets a new position in the list of resources.

'More' Select this gadget and you get a new window with more information about the selected resource.

'Functions'
All resource function offsets and addresses are shown up in a subwindow. If an appropriate '.fd' file exists and an 'FD:' assign points to its directory, then you will also see the function names. Note that some resources do not have functions.

'Exit' The 'Resources' window will be closed.

Please note: If you should find three dashes (minus signs) at 'OpenC' and/or 'RPC', the resource has no typical library structure. This happens for example at the 'FileSystem.resource'.

4.19 ScreenMode

Screen modes define all monitor resolutions that the system is able to show up.

You surely have already selected a screen mode before. In this list, you will find all available modes. Most of the programs filter out some of them when they do not meet their purpose.

Column items

'ModeID' An identification number, unique to each mode.

'Width' Nominal width of the resolution in pixeln.

'Height' Nominal height of the resolution in pixeln.

'Depth' Maximum number of planes. The amount of colors which can be simultaneously displayed depends on this.

'ScreenMode'
Name of the screen mode. Some modes do not have a real name, so Scout will generate it. Then it might differ from the name generated by other programs or screen mode requesters.

Actions

'Update' The list will be updated.

'Print' This function allows you to send the list to printer or a selected file.

'More' Further information about the screen mode are displayed in a subwindow. This includes the overscan resolutions and the frequencies. Due to an inaccuracy of the operating system, the real frequencies might be a little bit different to those displayed.

'Exit' The window will be closed.

4.20 Semaphores

The use of semaphores is a way of single-threading critical sections. For example only one program is allowed to use the printer at one time, otherwise the texts would be mixed.

Column items

'ln_Name' Name of a semaphore

'Nest' This item counts how often the semaphore has been obtained by the owner task.

'Queue' This counter shows you, how many programs want to obtain the semaphore.

'Owner' Here you will find the name of the task that owns the semaphore.

Actions

'Update' The list of 'Semaphores' will be updated.

'Print' This function allows you to send the list of 'Semaphores' to printer or a selected file.

'Obtain' This function is used to gain access to a semaphore. The 'NestCnt' will be increased at one by this call.

'Release' Herewith you can make a signal semaphore available to others.

'Exit' The 'Semaphores' window will be closed.

4.21 System

Actions

'`Print`' This function allows you to send the list to printer or a selected file.

'`Exit`' The window will be closed.

Entry items

In this window you will find some (partially technical) information about your computer. Please excuse the ordinary look of the window, but it is very easy to add more lines this way.

4.22 Tasks

In this window you find a list of all tasks and processes being in system. Each program you start will be executed as a task or process.

Column items

'`ln_Name`' Name of the task/process

'`ln_Type`' Type of the structure ('`task`' or '`process`')

'`ln_Pri`' Priority of the task/process

'`NUM`' If a non detaching program was started from shell, you will find here the number of the process. Programs you started from Workbench have a dash here.

'`State`' Here you see the state of the task or process. You will find Scout's own process on the top of the list with a '`run`' at this place, because this process is always running when it gets the task list.

'`ready`' means the task wants to work, but it's interrupted by the execution of another task.

A task that is waiting for a certain signal is in the state '`wait`'. In this case it doesn't need processing time.

'`SigWait`' Signalmask the task is waiting for.

Actions

'`Print`' This function allows you to send the list of '`Tasks`' to printer or a selected file.

'`Freeze`' With this function you freeze the selected task. It can still be found in the list of tasks, but it gets no processing time from system.

> **Warning:** If you try to freeze tasks essential to the system like 'input.device', you should have saved all important data, cause a RESET is the only way out!

'`Activate`'
A frozen task can be activated here.

'`CPU`' Here you will find a text field and a cycle gadget. This text field displays — dependent on the state of the cycle gadget — the CPU load in percent.

For the cycle gadget you can choose between three states:

'`off`' In this case the CPU load won't be displayed. If you select another state, `Scout` will patch some system functions to calculate the CPU load of all tasks.

'`full`' If you select this state, `Scout` sets the real cpu load to 100%. That means the total of the CPU loads of all tasks and processes will be 100%. Therefore nothing will be displayed in the text field.

'`in %`' `Scout` starts a task named ' `Scout's cheat task` ' to calculate the real CPU load and it will be displayed in the text field.

'`Secs`' This string gadget allows you to set the intervall time for updating of the CPU load display.

'`Update`' The list will be updated.

'`Remove`' A task will be removed from the list. You should prefer the freeze function, if you perhaps need this task again.

See also '`Break`'!

'`Signal`' If you select a signal mask, it will be send to the task.

'`Break`' A signal mask that includes the signals CTRL-C and CTRL-D will be send to the task you selected. Many tasks and processes end, if they receive these signals.

'`Priority`'
The priority of a task can be changed with this function.

'`More`' Selecting this gadget will open another window that displays more informations about the task or the process.

'`Exit`' The window will be closed.

4.23 Timer

This window lists all current requests of the timer.device.

Column items

'`Address`' Address of the IO request structure.

'`ReplyPort`'
Address of the port the request will be replied to.

'`Time`' Time that this request will stay in this list.

'`Unit`' There are two different measures: VBlank (which has less overhead) and MicroHz (which is more accurate).

'`Task`' Name of the requesting task.

Actions

'`Print`' This function allows you to send the list to printer or a selected file.

'`Update`' The list will be updated.

'`Exit`' The window will be closed.

4.24 Vectors

Actions

'`Update`' The displayed vectors will be updated.

'`Print`' This function allows you to send the list of '`Vectors`' to printer or a selected file.

'`Exit`' The window will be closed.

Reset Vectors

A program can make itself reset-protected by using the reset vectors. If the vectors are unused, they have a value of zero. The programs which use the Kick-Vectors (KickTagPtr, KickMemPtr and KickCheckSum) can also be found in the list of resident structures. See also Section 4.17 [Residents], page 23.

Auto Vector Interrupts

In a computer system with a MC68000 processor you will find the seven '`Auto Vector Interrupts`' from address $64 to address $7c. Higher processors (MC68010, etc.) have the VBR (Vector Base Register) that allows you to move the interrupt table to FAST-MEM. The system will be a little bit faster then. `Scout` uses the VBR if it exists.

Interrupt Vectors

Here you see 16 interrupt vectors (IntVecs). These vectors are located in the 'ExecBase' (base structure of the exec.library).

4.25 Windows

All screens with the windows opened on them are listed here. Screens are of a different color as windows.

Column items

'`Pos(x,y)`'
: x and y position of the screen/window

'`Size(x,y)`'
: x and y size of the screen/window

'`Title`'
: Title of the screen/window

Actions

'`Update`'
: The list will be updated.

'`Print`'
: This function allows you to send the list of '`Windows`' to printer or a selected file.

'`Close`'
: With this function it is possible to close screens and/or windows. If you close a screen, all windows on it will be closed too.

'`To Front`'
: The selected screen/window will be popped to front.

'`More`' If you select this gadget another window will be opened that displays more informations about the window or the screen.

'`Exit`' The window will be closed.

4.26 Patches

All patches currently installed in the system are listed here.

Column items

'`Library`' Name of the patched resource

'`Offset`' Decimal and Hex offset of the patched function

'`Function`'
Name of the patched function. This name is obtained from the accoring .fd file.

'`State`' Current state of the patch. This can either be '`active`', '`disabled`' or '`removed`'.

'`Patcher`' Name of the program that applied this patch.

Actions

'`Update`' The list will be updated.

'`Print`' This function allows you to send the list of '`Patches`' to printer or a selected file.

'`Enable`' If you select this gadget the selected (disabled) patches will be enabled again.

'`Disable`' If you select this gadget the selected (enabled) patches will be disabled.

> **Warning:** This is a very dangerous action!! Please read the SaferPatches documentation!

'`Exit`' The window will be closed.

4.27 Catalogs

All locale catalogs currently loaded are listed here.

Column items

'`cat_Version`'
: Version number of the catalog file.

'`cat_Language`'
: Language of the catalog file.

'`cat_Name`'
: Name of the catalog file.

Actions

'`Update`' The list will be updated.

'`Print`' This function allows you to send the list of '`Catalogs`' to printer or a selected file.

'`Exit`' The window will be closed.

4.28 AudioModes

All AHI audiomodes are listed here.

Column items

'`ID`' ModeID of the AudioMode.

'`Name`' Name of the AudioMode.

'`Bits`' Bits per sample.

'`min Freq`' Minimum mixing frequency.

'`max Freq`' Maximum mixing frequency.

Actions

'`Update`' The list will be updated.

'`Print`' This function allows you to send the list of '`Patches`' to printer or a selected file.

'`More`' Selecting this gadget will open another window that displays more informations about the AudioMode.

'`Exit`' The window will be closed.

4.29 Scout and AmiTCP

This section will show you what you have to do for using Scout as a TCP/IP service through AmiTCP. Nearly all functions of Scout can also be used via AmiTCP.

Now some knowledge will be assumed. If you don't know, what kind of program AmiTCP represents, you should read AmiTCP's user's manual before. (See also Section 3.4 [AmiTCP], page 7.)

If you have installed AmiTCP, you can use Scout as client and server. Except the installed programs of AmiTCP you don't need another program for using Scout on networks.

If you want to make your computer available for other systems on the network, you have to do following two steps:

1. Add the line '`scout 6543/tcp`' to file '`AmiTCP:db/services`'.
2. Now please add the line '`scout stream tcp nowait root dh0:scout`' to file '`AmiTCP:db/inetd.conf`'. Make sure that the path at the end of this line is the right path for `scout`.

That's it! If you start AmiTCP now, your computer is available for other systems through using the options '`HOST`', '`USER`' and '`PASSWORD`'.

> **Example:** If I want perform some actions on some system structures of my own system for example, I have to start Scout through something like:
>
> 1> scout HOST crash.north.de USER atte PASSWORD secret

If you leave out option '`PASSWORD`', you will be asked for the correct password through the '`password:`' prompt. In this case nobody can see your password, because it won't be displayed in shell.

If you don't use option '`USER`', AmiTCP takes the username that is actually available in system.

The usage of AmiTCP doesn't provide the installation of MUI. All of Scout's shell commands (see also Chapter 6 [Commands], page 37) can be used via network through AmiTCP.

> **Example:** If I want to get the task list of my system, I have to use something like:
>
> 1> scout HOST crash.north.de USER atte PASSWORD secret Tasks

You and all other users must always identify themselves through their usernames (option '`USER`') and their passwords (option '`PASSWORD`'). It's also possible to allow or deny certain systems the usage of some services through the file '`AmiTCP:db/inet.access`'. See also the user's manual of AmiTCP.

If you want to get more informations about the implemented options and commands, you should also see Chapter 5 [Options], page 35 and Chapter 6 [Commands], page 37.

NOTE: Please take care that the remote system runs the latest Scout version as well, to avoid incompatibilities.

4.30 Scout without MUI

Nearly all through the graphical user interface available functions of `Scout` are also available via shell. Therefore you don't really need `MUI` for using `Scout`. But if you want to use `Scout's` graphical user interface, you must have `MUI` in your system.

5 Options

There are some options for `Scout` which you can use, when you start the program. The following options are available from shell and as tool types from Workbench.

'`ICONIFIED`'

Usage: `ICONIFIED`

If this option is activ, `Scout` starts iconified.

'`PORTNAME`'

Usage: `PORTNAME`=*portname*

The name of Scout's ARexx port can be changed into *portname*. Without this option the ARexx port is called 'SCOUT.X'. The '`X`' stands for a decimal number that will be incremented, if a so called port already exists.

'`TOOLPRI`'

Usage: `TOOLPRI`=*value*

This option allows you to change the priority of Scout's process into *value*.

'`STARTUP`'

Usage: `STARTUP`=*command*

The variable *command* should be an ARexx script or a single ARexx command. Both (script or command) will be executed, when `Scout` will be started. In this way you can open more than only the main window by starting. Try for example the command '`OpenWindow Tasks`' and you will get two windows by starting (the main window and the task list window).

(See also Chapter 6 [ARexx port], page 37.)

'`INTERVALTIME`'

Usage: `INTERVALTIME`=*seconds*

This options allows you to save your preferred update time for the list of tasks. (See also Section 4.22 [Secs], page 27.)

'`CPUDISPLAY`'

Format: `CPUDISPLAY`=*value*

Through the variable *value* you can select the state of the '`CPU`' cycle gadget you find in the '`Tasks`' window. (See also Section 4.22 [CPU], page 27.)

- '`1`' means '`CPU: full`'
- '`2`' means '`CPU: in %`'

'HOST'

Format: HOST=*hostname*

This options allows you to specify the system (*hostname*) you want to manipulate via network through AmiTCP.

'USER'

Format: USER=*username*

You have to use this option to identify yourself by using Scout as a TCP/IP service.

'PASSWORD'

Format: PASSWORD=*password*

Without a password Scout can't connect to another system via network. This option allows you to set the correct password.

'COMMAND'

Format: COMMAND=*commandline*

Nearly all of Scout's implemented functions are available from shell through this option. You don't need the 'COMMAND' key to use this option. (See also Chapter 6 [Commands], page 37.)

'SINGLEWINDOWS'

Format: SINGLEWINDOWS

Some users don't like to handle the many windows of Scout. This option solves the problem of too many windows. If this option is selected, only one list window and only one detail window is opened at a time.

6 Scout's commands via ARexx and shell

Scout supports two kinds of commands:

1. commands only available from shell
2. commands available from ARexx and shell

ARexx port

It's a feature of MUI to give each application its own ARexx port. Therefore Scout also has an ARexx port that usually has the name 'SCOUT.X'. The 'X' stands for a decimal number that will be incremented, if a so called port already exists.

You will find the name of Scout's ARexx port in the window you get, if you select the 'Project/About' menu.

Using tasknames:

If a task or a process was started from shell and hasn't detached itself, you will find the name of the command being executed, where usually the taskname is displayed. The real name of those tasks usually is something like 'Background CLI', but such a taskname isn't useful.

Example: If you start a non detaching task like 'DH0:Debug/Sushi' from shell, you will see 'DH0:Debug/Sushi' as taskname.

Some ARexx commands need a taskname as parameter. You have to select those from CLI started self detaching tasks by using their command names like Scout displays them in the lists of tasks.

6.1 Commands only available from shell

'Help'

Format: Help

This command is the most important one and it doesn't need parameters. If you try Help, Scout prints a list of all available commands to shell. =:^)

Now 18 commands follow. These commands allow the user to get all lists of system structures from shell. Therefore you only need to install MUI for using Scout's graphical user interface.

Each of the following commands has a shortened form that stands behind the command in parentheses.

Allocations (a), BoopsiClasses (b), Commands (c), Devices (d), Timer (e), Fonts (f), Assigns (g), InputHandlers (h), Interrupts

(i), `LowMemory` (j), `Commodities` (k), `Libraries` (l), `Memory` (m), `Mounts` (n), `Locks` (o), `Ports` (p), `Residents` (r), `Semaphores` (s), `Tasks` (t), `Resources` (u), `Vectors` (v), `Windows` (w), `Expansions` (x), `System` (y) and `ScreenMode` (z).

Example: To get the list of ports, you only have to use '`scout ports`' or '`scout p`' from shell.

6.2 Commands available from ARexx and shell

'`FindTask`'

Usage: `FindTask` *task*

This command allows you to check, if task *task* exists in system or not. The result is the address of the task *task*, if it has been found. *task* can be the name or the address of a task.

'`FreezeTask`'

Usage: `FreezeTask` *task*

The task *taskname* will be frozen. After that it will still be found in system's task list, but then it doesn't need processing time. You can choose the name or the address of a task for *task*.

'`ActivateTask`'

Usage: `ActivateTask` *task*

If task *task* was frozen, it will be activated, otherwise an error occured. *task* is again a task's name or an address.

'`RemoveTask`'

Usage: `RemoveTask` *task*

This command removes the task *task*. It's lost forever.

'`BreakTask`'

Usage: `BreakTask` *task*

`Scout` sends the task *task* a certain signal mask that includes the signals CTRL-C and CTRL-D. Many programs support these signals and finish themselves, if they receive one of them.

'`SignalTask`'

Usage: `SignalTask` *task hexsignal*

This command allows you to send a signal *hexsignal* to the task *task*. The signal must specified as a hexadecimal number.

Example:

```
SendSignal 'scout' 0x001000
```

sends task '`scout`' a CTRL-C and after that `Scout` ends.

'`SetTaskPri`'

Usage: `SetTaskPri` *task priority*

The task *task* gets a new priority (*priority*).

'`RemovePort`'

Usage: `RemovePort` *port*

The port *port* will be removed from `Scout`. *port* can be the name of a port or its address.

'`GetLockNumber`'

Usage: `GetLockNumber` *lockpattern*

This command returns the number of locks which have paths matching to the pattern *lockpattern*.

Example: Use the command

```
GetLockNumber 'WORK:Utilities/#?'
```

and you will know, how many locks are currently used for files in the directory '`WORK:Utilities/`'.

'`RemoveLocks`'

Usage: `RemoveLocks` *lockpattern*

Use this command and all locks which have paths matching to the pattern *lockpattern* will be removed. (See also `GetLockNumber`.)

'`RemoveLock`'

Format: `RemoveLock` *lockaddress*

The lock at adress *lockaddress* will be removed.

'`FindNode`'

Usage: `FindNode` *nodetype nodename*

This command allows you to find a certain node. You only have to know its name (*nodename*) and its type (*nodetype*).

Nodetype can have following values: '`LIBRARY`', '`DEVICE`', '`RESOURCE`', '`MEMORY`', '`SEMAPHORE`', '`PORT`' or '`INPUTHANDLER`'.

Example: If you want to get the address of the '`disk.resource`' you must use:

```
FindNode RESOURCE 'disk.resource'
```

'`GetPriority`'

Usage: `GetPriority` *nodeaddress*

This command allows you to check the priority of a certain node structure. This includes all following structure types: tasks, libraries, devices, resources, ports, residents, input handlers, interrupts, semaphores and the elements of the memory list.

You only have to know the address (*nodeaddress*) of that structure.

Example: The following ARexx commands store the priority of your chip memory in the variable 'pri':

```
FindName MEMORY 'chip memory'
addr = result
GetPriority addr
pri = result
```

'SetPriority'

Usage: SetPriority *nodetype nodename*

If you want to change the priority of the node *nodename*, you can use this command. Again *nodetype* can have following values: 'LIBRARY', 'DEVICE', 'RESOURCE', 'MEMORY', 'SEMAPHORE', 'PORT' or 'INPUTHANDLER'.

'CloseLibrary'

Format: CloseLibrary *library*

The library *library* will be closed once. *library* can be the name of the library or its address.

'RemoveLibrary'

Format: RemoveLibrary *library*

The library *library* will be removed, if no program uses it.

'RemoveDevice'

Format: RemoveDevice *device*

The selected device *device* will be removed. For *device* use the name or the address of the device.

'RemoveResource'

Format: RemoveResource *resource*

The resource *resource* will be removed.

'ObtainSemaphore'

Format: ObtainSemaphore *semaphore*

This command allows you to obtain the given semaphore. *semaphore* can be the semaphore's name or address.

'ReleaseSemaphore'

Format: ReleaseSemaphore *semaphore*

The semaphore *semaphore* will be once released.

'RemoveSemaphore'

Format: RemoveSemaphore *semaphore*

You are able to remove the semaphore *semaphore* by using this command.

'RemoveInputhandler'

Format: `RemoveInputhandler` *inputhandler*

The input handler *inputhandler* selected through name or address will be removed.

'`FindResident`'

Usage: `FindResident` *resident*

This command returns the address of the resident structure *resident*.

'`FindInterrupt`'

Usage: `FindInterrupt` *interruptname*

The address of the interrupt *interruptname* will be returned.

'`RemoveInterrupt`'

Format: `RemoveInterrupt` *interruptname*

The interrupt you have selected through *interruptname* will be removed.

'`FlushDevs`'

Usage: `FlushDevs`

All not used devices will be removed. The used memory will be freed.

'`FlushFonts`'

Usage: `FlushFonts`

If a diskfont is in memory, but no program uses it, it will be removed.

'`FlushLibs`'

Usage: `FlushLibs`

All not used libraries will be removed. The used memory will be freed.

'`FlushAll`'

Usage: `FlushAll`

This function includes `FlushDevs`, `FlushFonts` and `FlushLibs`. All not used devices, libraries and fonts will be removed and the used memory will be freed.

'`ClearResetVectors`'

Usage: `ClearResetVectors`

The six reset vectors will be cleared, if you select this function (see Section 4.24 [Vectors], page 29).

'`PopToFront`'

Usage: `PopToFront` *title*

This command allows you to pop a screen or window to front. You only have to know its (*title*).

'`CloseWindow`'

Usage: `CloseWindow` *windowtitle*

This command closes the window that is specified through its title (*windowtitle*).

'`CloseScreen`'

Usage: `CloseScreen` *screentitle*

If you select this command, the screen (*screentitle*) will be closed with all its windows.

'`CloseFont`'

Format: `CloseFont` *address*

The font at address *address* will be closed once.

'`RemoveFont`'

Format: `RemoveFont` *address*

This command removes the font at address *address*, if it's not used by any program.

'`RemoveCommand`'

Format: `RemoveCommand` *address*

`Scout` makes the resident command at address *address* not resident.

'`RemoveAssign`'

Format: `RemoveAssign` *name*

With this command you're able to remove the assign *name*.

'`RemoveAssignList`'

Format: `RemoveAssignList` *name address*

This command removes the directory at address *address* from assign *name*. You will find the address of that directory in the list of assigns.

'`PrintList`'

Format: `PrintList` *listcharacter filename*

This command allows you to print a list (specified by the listcharacter) into the file *filename*.

Example:

```
PrintList t 'ram:tasklist'
```

will print the list of tasks into the file 'ram:tasklist'.

'`OpenWindow`'

Usage: `OpenWindow` *windowid*

All windows you get if you select a gadget of `Scout's` main window, can be opened with this command. The *windowid* is the same text you find on the main window gadgets.

Example:

```
OpenWindow 'Mounted Devs'
```

will open the window with the list of mounted devices.

'`CxAppear`'
'`CxDisappear`'
'`CxEnable`'
'`CxDisable`'
'`CxKill`'
'`CxListChg`'
'`CxUnique`'

Format: `Cx...` *name*

Sends the command to the commodity named *name*.

'`RemoveCx`'

Format: `RemoveCx` *commodity*

Removes the appropriate commodity from the list. Please consider this as 'emergency break'. Use it only if `CxKill` failed.

'`SetCxPri`'

Format: `SetCxPri` *commodity priority*

Sets the priority of a commodity.

'`RemoveClass`'

Format: `RemoveClass` *class*

The appropriate BOOPSI class is removed, if no objects and no subclasses are existing.

Appendix A

How to get updates

The newest version of `Scout` should always be available on AmiNet or Public Domain collections, which are up-to-date.

You can also find the latest version on my home page:

`http://www.is-koeln.de/einwohner/shred/`

Credits

Now we have to thank some people for supporting the development of `Scout` on many different kinds:

- Klaus 'gizmo' Weber, he was always available to Atte and his many questions (not a few) during the programming of `Scout`.
- Christian 'cosinus' Stelter, he gave the permission to use his many manuals.
- Stefan Stuntz for his great '`MagicUserInterface`'
- all (hopefully still) bug reporting and feature requesting people: Kai 'wusel' Siering, Martin Hauner, Peter Meyer, Karl 'Charly' Skibinski, Michael 'Mick' Hohmann, Thore Böckelmann, Bernardo Innocenti, Daniel Lundberg, . . .

  and last but not least
- all the others we've forgotten for reporting bugs, sending expansion boards data and so on.

How to reach the author

If you have questions, suggestions, bug reports or anything else, you can contact me at:

Richard Körber
Hornstraße 20
51465 Bergisch Gladbach
- Germany -

E-Mail: shred@chessy.aworld.de
richard.koerber@koeln.netsurf.de

Send E-Mails whenever possible..
If you want to contact Andreas Gelhausen, you can reach him at:

```
Andreas Gelhausen
Graf Spee Str. 23b
26123 Oldenburg
   - Germany -

E-Mail: atte@crash.north.de
```

Please do not contact him for bug reports, suggestions and similar. But if you feel the urge to send a gift, then he is the right address!
That's it! =:^)

Index

P

R

S

T

U

V

W

Table of Contents