Scout

System Monitor
Scout 37.42 (Release 1.3)
Edition 1.3
September 1994

von Andreas Gelhausen

1 Einleitung

Was ist Scout?

`Scout` ist ein Systemmonitor, d.h. viele für den reibungslosen Betrieb des Rechners notwendige Strukturen — wie z.B. Tasks, Ports, Assigns, System-Erweiterungen, residente Befehle, Interrupts, usw. — können angeschaut und auf einige dieser Strukturen können auch bestimmte Aktionen ausgeführt werden.

Es können zum Beispiel Tasks und Prozesse eingefroren, Windows und Screens geschlossen, Semaphore freigegeben und Interrupts aus dem System entfernt werden.

Viele dieser 'Aktionen' können auch mittels eines ARexx-Ports benutzt werden.

Allerdings ist dieses Programm an erster Stelle etwas für Programmierer und Tüftler — Leute, die einen etwas tieferen Einblick in bestimmte Strukturen haben möchten. Ein 'normaler' Benutzer wird mit dem Programm wohl nicht allzuviel anfangen können.

2 Rechtliche Dinge

Copyright

Keine Garantie

Es wird nicht garantiert, daß das Programm fehlerfrei ist und immer ordnungsgemäß arbeitet. Sie benutzen es auf eigene Gefahr. Für Folgeschäden, gleich welcher Art, die durch die Benutzung des Programmes `Scout` entstehen, übernimmt der Autor keine Haftung.

Giftware

`Scout` 37.42 ist *Giftware* und darf frei kopiert und benutzt werden. Wenn Ihnen das Programm gefällt und Sie es auch benutzen, würde ich mich allerdings sehr freuen, wenn Sie mir den Aufwand, den ich bei der Erstellung von `Scout` gehabt habe, etwas versüßen würden. Für kleine Geschenke jeglicher Art stehe ich immer gern zur Verfügung. =:^)

3 Vor dem Programmstart

Was Ihr System haben sollte

`Scout` benötigt mindestens die Kickstart Version 2.04 und die MUI Version 2.1. Siehe unten.

MUI - MagicUserInterface

MUI ist ein System zum Erzeugen und Unterstützen von grafischen Benutzungsoberflächen. Mit der Hilfe eines Konfigurationsprogrammes bekommt der Benutzer einer MUI-Applikation die Möglichkeit das Aussehen dieser Applikation seinem Geschmack anzupassen.

MUI wird als Shareware vertrieben. Um ein vollständiges Programmpaket zu bekommen, das viele Beispiele und mehr Informationen über die Registrierung beinhaltet, sollten Sie auf lokalen Bulletin Boards oder Public Domain Disketten nach einem File namens '`muiXXusr.lha`' Ausschau halten (XX steht für die letzte Versionsnummer).

Sie können sich auch direkt registrieren lassen, indem Sie 30.- DM oder 20.- US$ an die folgende Adresse schicken:

```
Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY
```

Installation

Für die Installation von `Scout` reicht es aus, nur das Programm `Scout` selbst und die Datei `scout.data` in ein Verzeichnis Ihrer Wahl zu kopieren. Danach können Sie es sofort starten. `Scout.data` beinhaltet Daten von System-Erweiterungen.

4 Wie wird Scout benutzt?

Wenn Sie das Programm starten, wird das Hauptfenster geöffnet, welches viele Gadgets beinhaltet. Jedes dieser Gadgets steht für eine bestimmte Art von für das Betriebssystem notwendigen Strukturen.

Sie können wählen zwischen:

Assigns, Devices, Expansions, Fonts, InputHandlers, Interrupts, Libraries, Locks, Memory, Mounted Devices, Ports, Resident Commands, Residents, Resources, Semaphores, Tasks, Vectors and Windows.

Betätigen Sie eines dieser Gadgets, dann wird ein weiteres Fenster geöffnet, welches die jeweils dazugehörende Liste von Strukturen beinhaltet.

Beispiel: Betätigen Sie das '`Tasks`'-Gadget, so wird ein Fenster mit der aktuellen Task-Liste des Systems geöffnet.

Diese ganzen Funktionen können auch jeweils über das Menu und durch eine Taste aufgerufen werden, die durch das unterstrichene Zeichen auf jedem Gadget bestimmt wird.

Mit diesem Programm können Sie auf viele dieser Strukturen bestimmte Aktionen ausführen lassen. Sollten Sie so etwas in Betracht ziehen, dann sollten Sie sich bewußt sein, was Sie tun.

> **Achtung:** Unsachgemäße Manipulation der System-Strukturen kann zum Absturz des Systems führen. In schweren Fällen kann dies einen Datenverlust zur Folge haben.

Hinweis: Da es für die Anleitung eines solchen Programmes zu aufwendig wäre, die angegebenen Strukturen bis ins letzte Detail zu erklären, wundern Sie sich bitte nicht, daß einige Detail-Informationen fehlen.

Da über diese Dinge schon Bücher über Bücher geschrieben wurden, verweise ich an dieser Stelle auf die dafür vorgesehene Fachliteratur!

4.1 Assigns

Ein Assign weist einem Verzeichnis einen logischen Namen zu.

Wenn Sie zum Beispiel einem Verzeichnis '`DH0:Daten/Dokumente`' den logischen Namen '`Texte:`' zuweisen, dann können Sie auf eine Datei *Dateiname*, die sich in diesem Verzeichnis befindet, auch durch die Angabe von 'Texte:*Dateiname*' zugreifen.

Spalteneinträge

'`Address`' An dieser Adresse beginnt die Struktur eines Assign-Eintrages.

'`Name`' Logischer Name eines Verzeichnisses oder Gerätes

'`Path`' Hier steht der Pfad des Verzeichnisses.

Aktionen

'`Update`' Betätigen Sie dieses Gadget, dann wird die Liste erneut eingelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Assigns`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Mit dieser Funktion wird der ausgewählte Assign-Eintrag aus dem System entfernt.

'`Exit`' Das '`Assigns`'-Fenster wird geschlossen.

4.2 Devices

Ein Device, das sich in dieser Liste befindet, ist — wie auch eine Library (siehe Abschnitt 4.7 [Libraries], Seite 12) — eine Ansammlung von Funktionen bzw. Routinen, denen bestimmte Aufgaben zugedacht wurden.

Das 'trackdisk.device' zum Beispiel beinhaltet Funktionen für die Handhabung von Disketten bzw. der Laufwerke.

Spalteneinträge

'`Address`' Adresse der Device-Struktur

'`ln_Name`' Name eines Devices

'`ln_Pri`' Priorität eines Devices

'`OpenC`' Zähler, der angibt, wie oft das Device geöffnet wurde.

'`RPC`' '`RPC`' steht für '`RAM Pointer Count`' und gibt an, wieviele Sprungadressen des Devices ins RAM zeigen. So eine ins RAM zeigende Einsprungadresse weist auf ein Programm (z.B. den `SetPatch`-Befehl) hin, welches die 'alte' Funktion verbessern bzw. erneuern möchte, indem es einfach die Sprungadresse der Funktion durch die Adresse einer eigenen Funktion ersetzt.

Viele Viren hängen sich auf diese Weise ins System. Diese Tatsache soll Sie aber jetzt nicht in Panik versetzen, da es sich in den meisten Fällen um kleine Patch-Programme — wie den `SetPatch`-Befehl von Commodore — handelt.

Sollten alle Sprungadressen eines Devices ins RAM zeigen, dann hat es seinen Programmcode im RAM stehen. Ein solcher '`RPC`'-Eintrag besteht aus drei Sternen, da es in dem Fall unwichtig ist, wieviele Sprungadressen ins RAM zeigen.

'`ln_Type`' Typ dieser Struktur (Hier sollte normalerweise '`device`' stehen.)

Aktionen

'`Update`' Die Device-Liste wird erneut ausgelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Devices`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Mit dieser Funktion wird das ausgewählte Device entfernt. Voraussetzung hierfür ist allerdings, daß es von keinem Programm mehr benutzt wird bzw. der '`OpenC`' gleich Null ist.

'`Priority`'
Die Prioritat des Devices kann hier von Ihnen verändert werden. Hierzu erscheint ein kleines Fenster, in dem Sie eine neue Priorität angeben können. Durch die veränderte Priorität bekommt das Device eventuell einen neuen Platz in der Device-Liste.

'`More`' Ein zusätzliches Fenster wird geöffnet, in dem Sie weitere Details des selektierten Devices finden.

Sie erreichen dasselbe, indem Sie einfach einen Doppelklick auf den jeweiligen Device-Eintrag ausführen.

'`Exit`' Das '`Devices`'-Fenster wird geschlossen.

4.3 Expansions (System-Erweiterungen)

In dieser Liste werden alle Ihre System-Erweiterungen angezeigt, die zur Zeit dem System zur Verfügung stehen (Grafikkarten, Speichererweiterungen usw.).

Spalteneinträge

'`BoardAddr`'
: Das ROM der Karte ist ab dieser Adresse im Speicher zu finden. Sollte es sich bei der Karte um eine Speichererweiterung handeln, ist hier die Anfangsadresse des konfigurierten Speichersegmentes zu finden.

'`BoardSize`'
: Handelt es sich bei dem Listen-Eintrag um eine Speichererweiterung, dann steht hier die Byte-Anzahl, die dem System durch diese Karte als Speicher zur Verfügung gestellt wird.

  Bei 'normalen' Karten wird hier nur die Größe des zur Karte gehörenden ROMs angegeben.

'`Manufacturer`'
: Herstellernummer, die von Commodore vergeben wird.

'`Product`' Produktnummer, die der System-Erweiterung vom Hersteller gegeben wird.

'`Serial#`' Seriennummer der Karte (Dieser Eintrag wird von den meisten Karten nicht benutzt.)

Aktionen

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Expansions`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`More`' Beim Betätigen dieses Gadgets erhalten Sie mehr Informationen über die selektierte System-Erweiterung in einem zusätzlichen Fenster.

Sie erreichen dasselbe, indem Sie einfach einen Doppelklick auf den jeweiligen Eintrag der Liste ausführen.

'`Exit`' Das '`Expansions`'-Fenster wird geschlossen.

Unbekannte System-Erweiterungen

Wenn Sie eine System-Erweiterung durch einfaches Anklicken des jeweiligen Eintrages mit der Maus selektieren, dann erhalten Sie den Namen der Herstellerfirma und die Bezeichnung der Karte in dem dafür vorgesehenen Textfeld unterhalb der Liste. Das passiert natürlich nur, sofern mir diese Daten bei der Erstellung der jeweiligen Programmversion von `Scout` bekannt waren!

Sollten diese Angaben fehlen oder nicht mit den Daten Ihrer System-Erweiterungen übereinstimmen, so möchte ich Sie bitten, mir die folgenden Daten zuzusenden, damit ich sie dem Programm beifügen bzw. sie korrigieren kann. In der nächsten Version der Datei `Scout.data` sollten diese Angaben dann vorhanden sein.

Daten zur Erfassung einer nicht namentlich genannten Erweiterung:

1. Herstellernummer (Manufacturer)
2. Produktnummer (Product)
3. Name des Herstellers
4. Bezeichnung der Hardware

Seien Sie hierbei bitte so genau wie möglich. Die Version der Erweiterung oder auch noch andere Angaben können hierbei nicht schaden.

4.4 Fonts

Alle Zeichensätze, die sich zur Zeit im System befinden bzw. von Programmen benutzt werden, sind in dieser Liste zu finden.

Spalteneinträge

'`YSize`' Vertikale Größe des Zeichensatzes

'`Count`' Zähler, der angibt, von wievielen Programmen der Zeichensatz gerade benutzt wird.

'`Type`' Steht an dieser Stelle '`ROMFONT`', so befindet sich dieser Zeichensatz im ROM. Bei '`DISKFONT`' wurde er von Diskette bzw. Festplatte geladen.

'`Name`' Name des Zeichensatzes

Aktionen

'`Update`' Die Liste der Zeichensätze wird aktualisiert.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Fonts`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Close`' Hiermit kann ein Zeichensatz geschlossen werden. '`Count`' verringert sich dann um eins.

'`Remove`' Mit dieser Funktion kann ein Zeichensatz aus dem System (Speicher) entfernt werden, vorausgesetzt er wird von keinem Programm mehr benötigt und befindet sich nicht im ROM.

'`Exit`' Das '`Fonts`'-Fenster wird geschlossen.

4.5 Inputhandler

Inputhandler kümmern sich um die Benutzereingaben, die im System ankommen (Tastendrücke, Mausklicks, usw.). Sie stehen wie an einem Fließband in einer Reihe und werten diese Eingaben aus. Der Inputhandler mit der höchsten Piorität bearbeitet diese Eingaben zuerst. Kann er mit den Eingaben nichts anfangen, reicht er sie in der Regel an den nächsten Inputhandler weiter.

Das System benutzt normalerweise für seinen Inputhandler die Priorität 50. Möchte also ein Inputhandler die Benutzereingaben vor dem System bekommen, braucht er eine höhere Priorität.

Spalteneinträge

'`ln_Name`' Name des Inputhandlers

'`ln_Pri`' Priorität des Inputhandlers

'`is_Data`' Ab dieser Adresse sind die Daten des Inputhandlers im Speicher zu finden.

'`is_Code`' Diese Adresse zeigt zum Programmcode des Inputhandlers. Sollte diese Adresse ins RAM zeigen, so wird sie andersfarbig dargestellt. Der Inputhandler des Betriebssystems hat seinen Programmcode im ROM.

Ein paar Viren klinken sich als Inputhandler ins System. Bei denen zeigt dann auch die '`is_Code`'-Adresse ins RAM. Wiederum gilt auch in einem solchen Fall: Nicht gleich die Panik bekommen, es gibt genug 'normale' Programme, die so verfahren.

Aktionen

'`Update`' Die Liste der Inputhandler wird auf den neuesten Stand gebracht.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`InputHandlers`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Ein Inputhandler kann mit Hilfe dieser Funktion aus dem System entfernt werden. Hierbei zieht man dem System aber eventuell den Stuhl unter dem Hintern weg. Das System kann dabei leicht abstürzen!

'`Priority`'
Die Priorität des Inputhandlers kann auf einen bestimmten Wert gesetzt werden. Wird die Priorität eines Inputhandlers verringert, kann es passieren, daß Programme nicht mehr auf bestimmte Dinge (z.B. das Drücken einer bestimmten Taste) reagieren, da ein Inputhandler mit einer höheren Priorität diese absorbiert.

Auch diese Liste wird vom System nach den Prioritäten sortiert. Ändern Sie also die Priorität eines Inputhandlers, dann bekommt dieser eventuell einen neuen Platz in der Liste.

'`Exit`' Das Fenster wird geschlossen.

4.6 Interrupts

Interrupts sind bestimmte Ereignisse, auf die das Betriebssystem reagieren muß. Für jeden Interrupt-Typ stehen meist sogar 'mehrere' Interrupt-Routinen zur Verfügung. Diese Interrupt-Routinen werden in einer Liste nach Prioritäten sortiert.

Sobald also ein bestimmter Interrupt auftritt, wird das laufende Programm solange unterbrochen, bis die zum jeweiligen Interrupt gehörende Liste der Interrupt-Routinen abgearbeitet wurde.

Spalteneinträge

'`ln_Name`' Diesem Text kann normalerweise entnommen werden, von welchem Programm die Interrupt-Routine installiert wurde und auch benötigt wird.

'`ln_Pri`' Priorität der Interrupt-Routine

'`is_Data`' Ab dieser Adresse sind im Speicher Daten zu finden, die zur Interrupt-Routine gehören.

'`is_Code`' Der Programmcode der Interrupt-Routine ist hier zu finden. Sollte diese Adresse ins RAM zeigen, so wird sie andersfarbig dargestellt.

'`NUM`' Diese Nummer beschreibt das Ereignis, bei dem die Interrupt-Routine aufgerufen wird. Eine kleine Information hierzu finden Sie im '`IntName`'-Eintrag des Interrupt-Detail-Fensters, das durch das Betätigen des '`More`'-Gadgets geöffnet wird.

Beispiel: Nummer 5 bedeutet, daß die Interrupt-Routine bei jedem neuen Bildaufbau ihres Monitors aufgerufen wird, was bei einem 50 Hz Monitor 50 mal in der Sekunde passiert. ('`VERTB (vertical blank interval)`')

Aktionen

'`Update`' Die Liste der Interrupt-Routinen wird aktualisiert.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der Interrupt-Routinen zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'Remove' Mit dieser Funktion kann eine Interrupt-Routine aus der Liste entfernt werden. Sollte es sich bei der Interrupt-Routine allerdings um einen Interrupt-Handler handeln, kann Scout diese Aktionen nicht ausführen. Ist dies der Fall, dann steht in der Spalte 'IntType' der Text 'Handler'.

Bei den Interrupt-Handlern vom audio.device kann dieses 'Problem' z.B. gelöst werden, indem das audio.device entfernt wird. Das passiert unter anderem durch den Aufruf von avail flush, wenn das audio.device von keinem Programm mehr benutzt wird.

'More' Ein Fenster mit weiteren Informationen über den selektierten Interrupt wird geöffnet.

'Exit' Betätigen Sie dieses Gadget, dann wird das Fenster geschlossen.

4.7 Libraries

Eine Library ist eine Ansammlung von Funktionen/Routinen (Bibliothek), denen bestimmte Aufgaben zugedacht wurden.

Die 'graphics.library' zum Beispiel beinhaltet Funktionen für die Grafikdarstellung.

Spalteneinträge

'Address' Adresse einer Library

'ln_Name' Name einer Library

'ln_Pri' Priorität einer Library

'OpenC' Zähler, der angibt, wie oft die Library geöffnet wurde.

'RPC' 'RPC' steht für 'RAM Pointer Count' und gibt an, wieviele Sprungadressen der Library ins RAM zeigen. So eine ins RAM zeigende Einsprungadresse weist auf ein Programm (z.B. den SetPatch-Befehl) hin, welches die 'alte' Funktion verbessern bzw. erneuern möchte, indem es einfach die Sprungadresse der Funktion durch die Adresse einer eigenen Funktion ersetzt.

Viele Viren hängen sich auf diese Weise ins System. Diese Tatsache soll Sie aber jetzt nicht in Panik versetzen, da es sich in den meisten Fällen um kleine Patch-Programme — wie den SetPatch-Befehl von Commodore — handelt.

Sollten alle Sprungadressen einer Library ins RAM zeigen, dann hat sie ihren Programmcode im RAM stehen. Ein solcher 'RPC'-Eintrag besteht aus drei Sternen, da es in dem Fall unwichtig ist, wieviele Sprungadressen ins RAM zeigen.

'ln_Type' Typ dieser Struktur (Hier sollte normalerweise 'library' stehen.)

Aktionen

'Update' Die Library-Liste wird erneuert.

'Print' Mit Hilfe dieser Funktion können Sie die Liste der 'Libraries' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'Remove' Mit dieser Funktion wird die selektierte Library entfernt. Voraussetzung hierfür ist allerdings, daß sie von keinem Programm mehr benutzt wird bzw. der 'OpenC' gleich Null ist.

Einige Libraries lassen sich nicht mehr ohne einen Reset aus dem System entfernen. Es ist also nicht unbedingt verwunderlich, wenn Scout es einmal nicht schaffen sollte, eine Library zu entfernen!

'Close' Um eine Library aus dem System entfernen zu können, muß sie von allen Programmen wieder geschlossen worden sein. Dies ist der Fall, wenn der 'OpenC'-Eintrag den Wert Null hat.

Wenn Sie mit dieser Funktion eine Library schließen möchten, werden Sie gefragt, ob Sie die Library nur einmal oder gleich für alle Programme schließen möchten, die diese Library geöffnet haben.

Wählen Sie hier also 'all', dann wird die Library so oft geschlossen, bis der 'OpenC' gleich Null ist.

'Priority'
Die Prioritat der Library kann von Ihnen verändert werden. Hierzu erscheint ein kleines Fenster, in dem Sie die neue Priorität angeben können. Durch die veränderte Priorität bekommt die Library eventuell einen neuen Platz in der Liste.

'More' Ein Fenster mit weiteren Informationen zur Library wird geöffnet.

'Exit' Das 'Libraries'-Fenster wird geschlossen.

4.8 Locks

Ein Lock symbolisiert den Zugriff eines Programmes auf eine Datei oder ein Verzeichnis. Auf diese Weise wird z.B. verhindert, daß eine Datei gelöscht wird, während irgendein anderes Programm noch auf die sich in der Datei befindenden Daten zugreift.

Bei etwas umfangreicheren Systemen kann der Aufbau der Liste etwas länger dauern! Mein eigenes System hat z.B. im Durchschnitt ca. 500 Lockeinträge, was gemessen an anderen Systemen noch nicht allzu viel ist. =:^)

Spalteneinträge

'Access' Hier wird die Zugriffsart des Lock-Zugriffes angegeben. Dies kann ein Lese- ('READ') oder ein Schreibzugriff ('WRITE') sein. Sollte hier 'OWN' stehen, dann handelt es sich nur um einen Lock, der zum Aufbau dieser Liste von Scout angefordert wurde.

'Path' Pfad der Datei oder des Verzeichnisses

Aktionen

'Update' Die Liste der Locks wird aktualisiert.

'Print' Mit Hilfe dieser Funktion können Sie die Liste der 'Locks' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'Remove' Ein Lock wird mittels der 'UnLock()'-Funktion der dos.library wieder freigegeben.

'Pattern' Geben Sie hier ein Namensmuster an, so werden nur die Locks angezeigt, deren Pfad mit dem Namensmuster übereinstimmt.

'Exit' Das Fenster wird geschlossen.

4.9 Memory (Speichersegmente)

Die Einträge dieser Liste stellen die Speichersegmente Ihres Rechners dar. Sie finden dort mindestens den Eintrag Ihres Grafik-Speichers ('CHIP-MEMORY'), der fest in Ihren Rechner eingebaut ist.

Spalteneinträge

'`ln_Name`' Name des Speichersegmentes (z.B. '`chip memory`')

'`ln_Pri`' Priorität des Speichersegmentes

'`mh_Lower`'
Anfangsadresse des Speichersegmentes

'`mh_Upper`'
Endadresse des Speichersegmentes

Aktionen

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der Speichersegmente zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Priority`'
Mit dieser Funktion können Sie bestimmen, welches Speichersegment bevorzugt vom System und den anderen Programmen benutzt werden soll, indem Sie diesem eine höhere Priorität geben als den anderen Speichersegmenten.

Ausnahme: Wird der Typ des Speichers direkt bei der Anforderung eines Programmes angegeben, wird das erste Speichersegment benutzt, das die Anforderungskriterien erfüllt.

'`More`' Ein neues Fenster wird geöffnet. Dieses Fenster enthält weitere Daten zum selektierten Speichersegment.

'`Exit`' Das '`Memory`'-Fenster wird geschlossen.

4.10 Mounted Devices

In dieser Liste finden Sie alle Ihre ansprechbaren Geräte (Laufwerke, Festplatten usw.).

Spalteneinträge

'`Name`' Name des Gerätes

'`Unit`' Kennziffer des Gerätes (Bei DF2: steht hier z.B. normalerweise eine Zwei.)

'`Heads`' Anzahl der vorhandenen Lese- bzw. Schreib-Köpfe

'`Cyl`' Anzahl der Zylinder

'`State`' Zustand eines Gerätes, der z.B. angibt, ob eine Diskette im Laufwerk liegt oder ob die Diskette unlesbar ist.

'`DiskType`'
Typ der Diskette (z.B. '`OFS`' (OldFileSystem), '`FFS`' (FastFileSystem), ...)

'`Handler or Device`'
Hier wird angegeben, welcher Handler oder welches Device sich um den Zugriff auf das jeweilige Gerät kümmert.

Beim Laufwerk DF0: wäre es z.B. in der Regel das 'trackdisk.device'. Um also direkt auf die Sektoren von DF0: schreiben zu können, müßten Sie das trackdisk.device benutzen.

Aktionen

'`Update`' Die Liste wird erneut eingelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der Geräte zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`More`' Ein weiteres Fenster mit mehr Informationen zum ausgewählten Gerät wird geöffnet.

'`Exit`' Das Fenster wird geschlossen.

4.11 Ports

Ports dienen der Kommunikation von Programmen. Dem Port eines Programmes können Mitteilungen gesendet werden, auf die das Programm reagieren soll.

Spalteneinträge

'`Address`' An dieser Adresse ist die Port-Struktur zu finden.

'`ln_Name`' Name des Ports

'`ln_Pri`' Priorität des Ports

'`mp_SigTask`'
Name des Tasks, der für diesen Port zuständig ist.

Aktionen

'`Update`' Die Liste der Ports wird aktualisiert.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Ports`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Der Port wird aus dem System entfernt.

'`Priority`'
Mit Hilfe dieser Funktion kann die Priorität des Ports verändert werden.

'`Exit`' Das '`Ports`'-Fenster wird geschlossen.

4.12 Resident Commands (Residente Befehle)

Alle Kommandos, die durch den Shell-Befehl `resident` resident gemacht wurden, und die Befehle, die schon im ROM enthalten sind, werden hier angezeigt.

Dabei werden auch die Positionen und die Größen aller Hunks der jeweiligen Befehle aufgelistet.

Die hier behandelten 'residenten Befehle' haben nichts mit den im nächsten Abschnitt beschriebenen 'residenten Strukturen' zu tun.

Spalteneinträge

'`Name`' Name des Befehls

'`UseCount`'
Zähler, der angibt, wieviele Instanzen des Befehls zur Zeit des Listenaufbaus im System aktiv sind.

'`Lower`' Startadresse eines Hunks im Speicher

'`Upper`' Endadresse eines Hunks im Speicher

'`Size`' Größe des Hunks ('`Upper`' - '`Lower`' - 8 Bytes Overhead)

Aktionen

'`Update`' Die Liste der residenten Befehle wird erneut eingelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der residenten Befehle zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Mit dieser Funktion wird der ausgewählte residente Befehl aus der Liste entfernt. Voraussetzung hierfür ist allerdings, daß er nicht mehr benutzt wird bzw. der '`UseCount`' gleich Null ist.

'`Exit`' Das Fenster wird geschlossen.

4.13 Residents (Residente Strukturen)

Residente Strukturen (Residents) sind Code- bzw. Daten-Segmente (wie zum Beispiel Libraries), die einen Reset überstehen. Sie sind *reset-fest*.

Die hier behandelten 'residenten Strukturen' haben nichts mit den im vorigen Abschnitt beschriebenen 'residenten Befehlen' zu tun.

Ein Programmierer hat nun die Möglichkeit sein Programm reset-fest zu machen, indem er unter anderem eine Resident-Struktur initialisiert und diese über die Kick-Vektoren (siehe Abschnitt 4.17 [Vectors], Seite 20), die sich in der ExecBase-Struktur (Basis der exec.library) befinden, ins System einklinkt.

Diese residenten Strukturen liegen demnach im RAM und ihre Adressen werden andersfarbig dargestellt, um sie von den anderen residenten Strukturen abzuheben. Die residenten Strukturen, die über die Kick-Vektoren ins System gekommen sind, werden, sofern überhaupt solche residenten Strukturen vorhanden sind, am oberen Ende der Liste eingefügt.

Sollten Sie hier eine residente Struktur finden, die ins RAM zeigt, dann ist Vorsicht geboten. Schauen Sie sich ihren Namen an, und wenn Sie nicht ganz sicher wissen, worum es sich handelt, sollten Sie lieber einmal den Virenkiller Ihres Vertrauens das System überprüfen lassen.

Viele Viren machen sich auf diese Weise reset-fest!

Spalteneinträge

'`Address`' An dieser Adresse ist die residente Struktur zu finden.

'`ln_Name`' Name der residenten Struktur

'`rt_Pri`' Priorität der residenten Struktur

'`rt_IdString`'
Identifikationstext der residenten Struktur

Aktionen

'`Update`' Die Liste der residenten Strukturen wird aktualisiert.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Residents`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`More`' Ein neues Fenster mit mehr Informationen über die Resident-Struktur wird geöffnet.

'`Exit`' Das '`Residents`'-Fenster wird geschlossen.

4.14 Resources (Resourcen)

Eine Resource ist — wie auch eine Library (siehe Abschnitt 4.7 [Libraries], Seite 12) und ein Device (siehe Abschnitt 4.2 [Devices], Seite 8) — eine Ansammlung von Funktionen bzw. Routinen, denen bestimmte Aufgaben zugedacht wurden.

Spalteneinträge

'`Address`' Addresse der Resource

'`ln_Name`' Name der Resource

'`ln_Pri`' Priorität der Resource

'`OpenC`' Zähler, der angibt, wie oft die Resource geöffnet wurde.

'`RPC`' '`RPC`' steht für '`RAM Pointer Count`' und gibt an, wieviele Sprungadressen der Resource ins RAM zeigen. So eine ins RAM zeigende Einsprungadresse weist auf ein Programm hin (wie z.B. den `SetPatch`-Befehl), welches die 'alte' Funktion verbessern bzw. erneuern möchte, indem es einfach die Sprungadresse der Funktion durch die Adresse einer eigenen Funktion ersetzt.

Sollten alle Sprungadressen einer Resource ins RAM zeigen, dann hat sie ihren Programmcode im RAM stehen. Ein solcher '`RPC`'-Eintrag besteht aus drei Sternen, da es in dem Fall unwichtig ist, wieviele Sprungadressen ins RAM zeigen.

'`ln_Type`' Typ der Struktur (Hier sollte normalerweise '`resource`' stehen.)

Aktionen

'`Update`' Die Resource-Liste wird neu eingelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der '`Resources`' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Remove`' Mit dieser Funktion wird die gewählte Resource entfernt. Voraussetzung hierfür ist allerdings, daß sie von keinem Programm mehr benutzt wird bzw. der '`OpenC`' gleich Null ist.

'`Priority`'
Die Priorität der Resource kann von Ihnen verändert werden. Hierzu erscheint ein kleines Fenster, in dem Sie die neue Priorität angeben können.

'`More`' Wird dieses Gadget betätigt, dann erscheint ein zusätzliches Fenster mit weiteren Daten zur selektierten Resource.

'`Exit`' Das '`Residents`'-Fenster wird geschlossen.

Beachte: Sollte bei '`OpenC`' und/oder '`RPC`' ein Strich stehen, so besitzt die Resource keine typische Library-Struktur (Hintereinanderreihung von Sprungbefehlen und deren Sprungadressen). Das passiert z.B. beim Eintrag der '`FileSystem.resource`'.

4.15 Semaphores (Semaphore)

Semaphore sind normalerweise dafür da, den Zugriff auf bestimmte Geräte zu handhaben, auf die nur eine bestimmte Anzahl von Programmen zur Zeit zugreifen darf.

Beispiele:

1. Auf einen Drucker darf nur ein Programm zur Zeit zugreifen, da sonst die zu druckenden Texte 'gemischt' würden.

2. Wenn der `SetPatch`-Befehl von Commodore z.B. schon die Routinen des Betriebssystems gepatcht hat, dann soll er diese Patches beim nächsten Aufruf ja nicht nochmal ausführen. Zu diesem Zweck wird ein Semaphor eingerichtet. Der `SetPatch`-Befehl kann dadurch bei einem erneuten Start prüfen, ob er schon einmal ausgeführt worden ist.

Spalteneinträge

'`ln_Name`' Name des Semaphors

'`NestCnt`' Zugriffszähler

'`ln_Type`' Typ der Struktur (Hier steht normalerweise '`signalsem`'.)

Aktionen

'`Update`' Die Liste der Semaphore wird erneut eingelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der Semaphore zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Obtain`' Hierdurch wird dem System vorgegaukelt, daß das Gerät, das File oder wofür der Semaphor sonst eingerichtet wurde, gerade benutzt wird. Der '`NestCnt`'-Eintrag erhöht sich hierbei um Eins.

'`Release`' Sollte ein Semaphor gerade benutzt werden, so machen Sie dem System mit dieser Funktion weis, daß dem nicht mehr so ist. Ein Programm, das den Semaphor beachtet, kann so eventuell versuchen, ein weiteres Mal auf das entsprechende Gerät zuzugreifen.

'`Exit`' Das '`Semaphores`'-Fenster wird geschlossen.

4.16 Tasks

In dieser Liste befinden sich alle Tasks und Prozesse. (Prozesse sind erweiterte Task-Strukturen.) Sie repräsentieren die Programme, die im Augenblick im System ablaufen bzw. auf ein Ereignis warten.

Spalteneinträge

'`ln_Name`' Name des Tasks

'`ln_Type`' Typ der Struktur ('`task`' oder '`process`')

'`ln_Pri`' Priorität des Tasks

'`NUM`' Hier steht die Nummer eines Prozesses, sofern dieser sich mit Hilfe des Befehles `run` abgekoppelt hat oder noch in einer Shell läuft. Ein Programm, das über die Workbench gestartet wurde, hat als '`NUM`'-Eintrag einen Strich, wie auch ein Programm, das sich selbständig von der Shell abgekoppelt hat.

'`State`' Dieser Eintrag zeigt den Zustand eines Tasks/Prozesses an. Der eigene Prozess von `Scout`, der ganz oben in der Liste zu finden ist, hat dort immer '`run`' stehen, weil er immer aktiv ist, wenn er die Task-Liste ausliest. =:^)

Ein '`wait`' bedeutet hierbei, daß ein Task auf ein bestimmtes Ereignis wartet. Dies kann zum Beispiel das Betätigen eines Gadgets sein.

Sollte sich ein Task im Zustand '`ready`' befinden, dann hat er zwar gerade etwas zu tun, wurde aber von der Abarbeitung eines anderen Prozesses unterbrochen (Multitasking-Prinzip).

'SigWait' Signalmaske, auf die der Task wartet. Sollte ein Task im Zustand 'wait' sein und diese Signalmaske den Wert Null ($00000000) haben, dann handelt es sich mit großer Wahrscheinlichkeit um einen Task, der sich 'aufgehängt' hat und vom Betriebssystem in der Schwebe gehalten wird. ('suspend' or 'reboot')

Aktionen

'Update' Die Liste der Tasks und Prozesse wird erneut eingelesen.

'Print' Mit Hilfe dieser Funktion können Sie die Liste der 'Tasks' zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'Remove' Ein Task wird aus der Liste entfernt. Sollten Sie sich nicht ganz sicher sein, ob Sie den Task noch einmal brauchen, dann sollten Sie lieber die 'Freeze'-Funktion benutzen. (Siehe auch 'Break'!)

'Freeze' Hiermit wird ein Task *eingefroren*. Er befindet sich zwar dann noch in der Task-Liste, bekommt aber keine Rechenzeit mehr vom System.

> **Achtung:** Wenn Sie versuchen Tasks einzufrieren, die für das System *lebenswichtig* sind (wie z.B. der Task 'input.device'), sollten Sie alle wichtigen Daten abgespeichert haben, da durch den folgenden Systemabsturz diese Daten sonst verloren sind.

'Activate'
Ein eingefrorener Task kann hiermit wieder aktiviert werden.

'Secs' Mit Hilfe dieses String-Gadgets können Sie bestimmen, in welchen Intervallen die CPU-Belastung durch einzelne Tasks gemessen wird, sofern Sie diese Funktion mittels 'CPU/No CPU' überhaupt ausgewählt haben. Dieses Intervall sollte nicht zu klein gewählt werden, da es zu Ungenauigkeiten kommen kann und Scout dann die meiste Zeit der CPU benötigt. Intervalle kleiner 0.5 Sekunden machen nicht viel Sinn!

'CPU/No CPU'
Dieses Cycle-Gadget erlaubt es Ihnen, die Berechnung der CPU-Belastung durch die einzelnen Tasks in Prozent darstellen zu lassen. Wird dieses Cycle-Gadget das erste Mal auf 'CPU' gestellt, dann werden ein paar Systempatches installiert, die zur Berechnung notwendig sind. Dies geschieht auf die gleiche Weise, wie der SetPatch-Befehl von Commodore arbeitet. Ein Semaphor wird angelegt, der angibt, daß der Patch installiert wurde. Wird Scout nun erneut aufgerufen, dann werden die bereits installierten Routinen zur Berechnung der CPU-Auslastung benutzt.

'Signal' Sie können beim Benutzen dieser Funktion eine Signalmaske angeben, die darauf dem ausgewählten Task geschickt wird.

'Break' Einem Task wird ein 'Break'-Signal gesendet. Viele Tasks reagieren auf dieses Signal und beenden sich selbst. Reagiert der Task, der mit Hilfe von Scout aus dem System entfernt werden soll, auf dieses Signal, dann sollte er normalerweise den von ihm angeforderten Speicher wieder freigeben. Wird ein Task durch die 'Remove'-Funktion entfernt, wird der von ihm benutzte Speicher nicht wieder freigegeben. Es bleiben dann sogenannte 'Speicherleichen' im System zurück.

'Priority'
Die Priorität eines Tasks kann hiermit verändert werden. Ein Task mit einer niedrigen Priorität bekommt erst vom System Rechenzeit zur Verfügung gestellt, wenn kein Task mit einer höheren Priorität Rechenzeit benötigt.

'More' Ein weiteres Fenster wird geöffnet, das, je nachdem ob ein Task oder ein Prozess selektiert wurde, weitere Informationen zu dem Task oder dem Prozess beinhaltet.

'Exit' Das Fenster mit der Task-Liste wird geschlossen.

4.17 Vectors (Spezielle Vektoren)

Aktionen

'`Update`' Die Vektoren werden erneut ausgelesen.

'`Print`' Mit Hilfe dieser Funktion können Sie die Liste der Vektoren zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

'`Exit`' Das '`Vectors`'-Fenster wird geschlossen.

Reset Vectors

Mit Hilfe der Reset-Vektoren kann sich ein Programm reset-fest ins System einhängen. Sie haben einen Wert von Null, wenn sie nicht verbogen wurden. Benutzt ein Programm die Kick-Vektoren (KickTagPtr, KickMemPtr und KickCheckSum) um sich reset-fest zu machen, dann ist es auch in der Liste der residenten Strukturen zu finden. Siehe auch Abschnitt 4.13 [Residents], Seite 16.

Auto Vector Interrupts

Die 7 Auto-Vektor-Interrupts, die hier angezeigt werden, sind bei einem System mit MC68000-Prozessor von Adresse $64 bis $7c zu finden. Die Prozessoren MC68010 und aufwärts besitzen ein Vektor-Basis-Register (VBR), das eine Verlegung der Interrupt-Tabelle ins FAST-RAM ermöglicht. Durch diese Verlegung ins FAST-RAM wird das System etwas beschleunigt. `Scout` berücksichtigt das VBR bei der Darstellung dieser Vektoren, vorausgesetzt es ist vorhanden und wird benutzt.

Interrupt Vectors

Die hier angezeigten 16 Interrupt-Vektoren (IntVecs) befinden sich in der ExecBase-Struktur (der Basisstruktur der exec.library). Welche Aufgabe sie haben bzw. wie das Zusammenspiel der Auto-Vektor-Interrupts, der Interrupt-Vektoren und der Interrupt-Handler bzw. Interrupt-Server (siehe Abschnitt 4.6 [Interrupts], Seite 11) funktioniert, entnehmen Sie bitte der Fachliteratur.

4.18 Windows (Fenster)

In dieser Liste werden alle Screens mit den auf ihnen befindlichen Fenstern angezeigt. Screens werden andersfarbig dargestellt, damit sie sich besser von den Fenstern unterscheiden.

Spalteneinträge

'`Pos(x,y)`'
Horizontale (X) und vertikale (Y) Position des Screens/Fensters

'`Size(x,y)`'
Horizontale (X) und vertikale (Y) Größe des Screens/Fensters

'`Title`' Titel des Screens/Fensters

Aktionen

‘`Update`’ Die Liste wird erneut eingelesen.

‘`Print`’ Mit Hilfe dieser Funktion können Sie die Liste der Fenster zum Drucker schicken oder in eine Datei Ihrer Wahl ausgeben lassen.

‘`Close`’ Ihnen wird hiermit die Möglichkeit gegeben, Fenster und Screens zu schließen. Ein Screen wird dann mit all den Fenstern geschlossen, die sich auf ihm befinden.

‘`More`’ Je nachdem, ob ein Screen oder ein Fenster in der Liste selektiert wurde, wird ein weiteres Fenster geöffnet, das weitere Daten zum Screen oder zum Fenster enthält.

‘`Exit`’ Das ‘`Windows`’-Fenster wird geschlossen.

5 Optionen

Für das Programm stehen ein paar Optionen zur Verfügung, die Sie benutzen können, wenn Sie das Programm starten. Diese Optionen können als Shell-Parameter oder als Tool Types von der Workbench benutzt werden.

Beispiel: Starten Sie `Scout` durch

```
Scout TOOLPRI=1 ICONIFIED
```

dann wird das Programm iconifiziert gestartet und bekommt die Task-Priorität 1.

'`ICONIFIED`'

Format: `ICONIFIED`

Wird diese Option verwendet, dann startet `Scout` iconifiziert.

'`PORTNAME`'

Format: `PORTNAME`=*portname*

Der ARexx-Port von `Scout` kann mit Hilfe dieser Option in *portname* umbenannt werden. Wird diese Option nicht benutzt, dann bekommt der ARexx-Port von `Scout` den Namen 'SCOUT.X', wobei das 'X' die Nummer der `Scout`-Inkarnation angibt.

'`TOOLPRI`'

Format: `TOOLPRI`=*value*

Diese Option erlaubt es Ihnen, die Task-Priorität von `Scout` auf einen bestimmten Wert *value* zu setzen. Dieser Wert *value* darf nur Werte von -128 bis 127 annehmen.

'`STARTUP`'

Format: `STARTUP`=*scriptname*

Benutzen Sie diese Option, dann wird das ARexx-Skript *scriptname* jedesmal ausgeführt, wenn `Scout` gestartet wird.

Auf diese Weise kann zum Beispiel bei jedem Start des Programmes das '`Tasks`'-Fenster automatisch geöffnet werden. Dafür braucht das ARexx-Skript nur den Befehl '`OpenWindow Tasks`' zu beinhalten.

6 Scouts ARexx-Schnittstelle

MUI gibt jeder seiner Applikationen automatisch eine ARexx-Port (ARexx-Schnittstelle). Demnach besitzt `Scout` also auch einen ARexx-Port, der normalerweise den Namen '`SCOUT.X`' hat, wobei das 'X' die Nummer der Programm-Inkarnation angibt.

Der jeweilige Name des ARexx-Ports jeder `Scout`-Inkarnation wird auch in dem Fenster angezeigt, welches Sie durch die Auswahl des '`Project/About`'-Menüpunktes erhalten.

Verwendung von Tasknamen:

Ein Task oder ein Prozess, der von einer Shell aus gestartet wurde und sich nicht abgekoppelt hat, hat meistens einen Namen wie '`Background CLI`' oder '`CLI Process`'. `Scout` verwendet in der Task-Liste in einem solchen Fall nicht den 'richtigen' Namen des Tasks, sondern den Namen des jeweils ausgeführten Programmes.

Beispiel: Starten Sie zum Beispiel das Programm `DH0:Debug/Sushi` ohne den Befehl `run`, dann wird bei `Scout` als Taskname '`DH0:Debug/Sushi`' angezeigt.

Einige ARexx-Befehle von `Scout` erwarten als Parameter auch einen Tasknamen. Dieser Taskname muß auf die gleiche Weise angegeben werden, wie er bei `Scout` angezeigt wird.

Scout unterstützt folgende ARexx-Kommandos:

'`FindTask`'

Format: `FindTask` *taskname*

Dieses Kommando gibt die Adresse des Tasks *taskname* zurück. Sollte dieser Task nicht im System vorhanden sein, so wird der Rückgabecode, der in der ARexx-Spezialvariablen *RC* zu finden ist, auf den Wert 5 gesetzt.

'`FreezeTask`'

Format: `FreezeTask` *taskname*

Der Task *taskname* wird von `Scout` eingefroren. Er ist danach zwar noch in der Task-Liste zu finden, bekommt aber keine Rechenzeit mehr vom System.

'`ActivateTask`'

Format: `ActivateTask` *taskname*

Der eingefrorener Task *taskname* kann hiermit wieder aktiviert werden.

'`RemoveTask`'

Format: `RemoveTask` *taskname*

Mit diesem Kommando wird der Task *taskname* aus dem System entfernt.

'`SendBreak`'

Format: `SendBreak` *taskname*

Dem Task *taskname* wird mit Hilfe dieses Kommandos ein Signal geschickt, das dem Drücken von CTRL-C bzw. CTRL-D entspricht. Viele Programme reagieren auf dieses Signal, indem sie sich selbständig beenden.

'`SendSignal`'

Format: `SendSignal` *taskname hexsignal*

Hiermit kann dem Task *taskname* ein gewähltes Signal *hexsignal* (bzw. eine Signalmaske) zugeschickt werden. Dieses Signal muß als Hexadezimal-Wert (mit vorangestelltem '`0x`') angegeben werden.

Beispiel: Das Kommando

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

sendet dem `Scout`-Prozess ein CTRL-C, worauf dieser sein Dasein beendet.

'`SetTaskPri`'

Format: `SetTaskPri` *taskname priority*

Der Task *taskname* bekommt mit Hilfe dieses Kommandos die Priorität *priority*.

'`RemovePort`'

Format: `RemovePort` *portname*

Der Port *portname* wird von `Scout` aus dem System entfernt.

'`GetLockNumber`'

Format: `GetLockNumber` *lockpattern*

Dieses Kommando gibt die Anzahl der Lock-Einträge zurück, deren Pfade mit dem Namensmuster *lockpattern* übereinstimmen. So kann über ARexx nachgeschaut werden, ob noch auf ein bestimmtes File zugegriffen wird.

'`RemoveLocks`'

Format: `RemoveLocks` *lockpattern*

Alle Locks werden aus dem System entfernt, deren Pfade mit dem Namensmuster *lockpattern* übereinstimmen. Bei diesem Kommando ist höchste Vorsicht geboten! Will ein Programm einen Lock entfernen, der schon von `Scout` entfernt wurde, dann stürzt mit großer Wahrscheinlichkeit der Rechner ab.

'`OpenWindow`'

Format: `OpenWindow` *windowid*

Mit diesem Kommando sind Sie in der Lage, alle Fenster über ARexx zu öffnen, die über das Hauptfenster von `Scout` durch das Betätigen eines Gadgets geöffnet werden können.

Die Fensteridentifikation *windowid* besteht aus dem gleichen Text, der auch auf den Gadgets im Hauptfenster zu finden ist.

Beispiel: Wird das Kommando

```
OpenWindow 'Resident Cmds'
```

zu Scouts ARexx-Port geschickt, dann wird das Fenster mit der Liste der residenten Befehle geöffnet.

Sollte das Fenster schon geöffnet worden sein, dann wird es nach vorn geholt, und die jeweilige Liste wird neu eingelesen.

'`FindName`'

Format: `FindName` *nodetyp nodename*

Dieses Kommando erlaubt es Ihnen, eine Struktur *nodename* zu finden, die einen bestimmten Nodetypen *nodetype* besitzt.

Die Variable *nodetype* kann folgende Werte haben: '`LIBRARY`', '`DEVICE`', '`RESOURCE`', '`MEMORY`', '`SEMAPHORE`', '`PORT`' oder '`INPUTHANDLER`'.

Beipiel: Wenn Sie die Adresse der 'dos.library' bekommen möchten, müssen Sie das Kommando

```
FindName LIBRARY 'dos.library'
```

benutzen.

'`GetPriority`'

Format: `GetPriority` *nodeaddress*

Dieses Kommando liefert die Priorität einer Struktur, die folgenden Typ haben kann: Task, Library, Device, Resource, Port, Resident, Inputhandler, Interrupt, Semaphor oder ein Element der Memory-List.

Die Struktur müssen Sie dabei durch ihre Adresse *nodeaddress* auswählen, die Sie z.B. durch das ARexx-Kommando `FindName` erhalten.

Beispiel: Die folgenden ARexx-Kommandos beschaffen die Priorität Ihres Grafik-Speichers und legen sie in der Variablen '`pri`' ab:

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

'`SetPriority`'
Format: `SetPriority` *nodetype nodename*

Wenn Sie die Priorität einer Struktur *nodename* ändern möchten, können Sie dafür dieses Kommando benutzen. Wiederum kann die Variable *nodetype* folgende Werte haben: '`LIBRARY`', '`DEVICE`', '`RESOURCE`', '`MEMORY`', '`SEMAPHORE`', '`PORT`' oder '`INPUTHANDLER`'.

'`FindResident`'
Format: `FindResident` *residentname*

Dieses Kommando liefert die Adresse der residenten Struktur *residentname*.

'`FindInterrupt`'
Format: `FindInterrupt` *interruptname*

Die Adresse des Interrupts *interruptname* wird mit Hilfe dieses Kommandos beschafft.

'`FlushDevs`'
Format: `FlushDevs`

Sollten sich noch Devices im System/im Speicher befinden, die im Augenblick von keinem Programm mehr benötigt werden, so werden sie aus dem Speicher entfernt.

'`FlushFonts`'
Format: `FlushFonts`

Unbenutzte Zeichensätze, die von Diskette/Festplatte nachgeladen wurden und nicht mehr benötigt werden, werden aus dem Speicher entfernt.

'`FlushLibs`'
Format: `FlushLibs`

Sollten sich noch Libraries im System/im Speicher befinden, die im Augenblick von keinem Programm mehr benötigt werden, so werden sie aus dem Speicher entfernt.

'`FlushAll`'
Format: `FlushAll`

Diese Funktion beinhaltet die Funktionen `FlushDevs`, `FlushFonts` und `FlushLibs`. Dementsprechend werden Devices, Libraries und Zeichensätze, die zur Zeit von keinem Programm benutzt werden, aus dem Speicher entfernt.

'`ClearResetVectors`'
Format: `ClearResetVectors`

Bei Gebrauch dieser Funktion werden die sechs Reset-Vektoren gelöscht (siehe auch Abschnitt 4.17 [Vectors], Seite 20).

Anhang A

Wie und wo bekommt man Updates?

Die neueste Version von `Scout` sollte immer in dem "DEEP THOUGHT BBS" (siehe unten), im AmiNet oder etwas später in aktuelleren Public Domain Sammlungen vorhanden sein.

Support BBS

```
DEEP THOUGHT Bulletin Board System, Oldenburg, Germany

Node 1
+49-(0)441-383365   1200-21600 bps  v.32terbo, v.42bis

Node 2
+49-(0)441-383839   1200-19200 bps  v.32bis, v.42bis, ZyXEL

                Node 1                Node 2
FidoNet         2:2426/2020.0         2:2426/2021.0
AmigaNet        39:170/204.0          39:170/205.0

InterNet        cosinus@deepthought.north.de


Beide Nodes sind 24 Stunden am Tag online und auf beiden Nodes läuft
ein FidoNet-Mailer, der Fido-File-Requests akzeptiert.

Benutzen Sie das Magic  SCOUT für die neueste Version von SCOUT
oder                    FILES für eine komplette Fileliste
```

Wem ich zu danken habe

Nun habe ich noch ein paar Leuten zu danken, als da wären:

Klaus ‘gizmo’ Weber, der dieses Programm ein wenig unter die Lupe genommen hat und für meine Probleme bei der Entwicklung von `Scout` (es waren nicht wenige) meist ein freies Ohr hatte

Christian ‘cosinus’ Stelter, der mir erlaubt hat, seine ganzen Manuals zu benutzen

Stefan Stuntz für sein ‘`MagicUserInterface`’

Kai ‘wusel’ Siering, für das stetige Testen von `Scout`

und zum guten Schluß

all den anderen, bisher nicht genannten Leuten, die mir Bugs, Anregungen und konstruktive Kritik zu Gehör gebracht haben.

Wie erreicht man den Autor?

Wenn Sie Fragen, Verbesserungsvorschläge, Bug Reports oder Dinge dieser Art haben, dann können Sie mich unter den folgenden EMail-Adressen erreichen:

```
atte@crash.north.de (Andreas Gelhausen)
                oder
      2:2426/2020.24 (im FidoNet)
```

Wenn Sie nicht über die Möglichkeit verfügen, mich über die oben angegebenen EMail-Adressen zu erreichen, dann können Sie mir natürlich auch 'normale' Briefe schreiben.

Hier meine Adresse:

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

Stichwortverzeichnis

Inhaltsverzeichnis