Sysmon

Etienne Vogt

COLLABORATORS			
	TITLE : Sysmon		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Etienne Vogt	August 24, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sysmon

1.1 Sysmon Guide

Sysmon Documentation V1.10 (sysmon.library 1.20)

1.2 Introduction

INTRODUCTION

Sysmon is a system monitoring and enhancing package based on a shared library and Shell based support commands. It requires AmigaOS 2.04 or higher. Installation of Richard Körber's Identify.library (util/libs/Identify.lha on Aminet) is recommanded if you want the Sysmon monitor to be able to identify expansion boards. If your Amiga has a MMU (Memory Management Unit), it is also recommanded, but not required, that you install Thomas Richter's mmu.library, available from Aminet (util/libs/MMULib.lha).

Some of the features provided by Sysmon are :

- Precise CPU usage timing for all tasks with very low overhead. - System message logging daemon like UNIX syslogd (useful for debugging). - Safe System shutdown script (useful for BBS's and networked machines). - Functions allowing to send/receive systemwide broadcast messages. - New Alert function (the Guru is back) allowing detailed logging. - New system functions

related to tasks, including an enhanced and bug fixed task signal exception mechanism. - Executive compatible SysInfo.library to query tasks CPU usage and load averages.

Sysmon has been developped and tested on the following configurations :

- A4000T 68060 AGA + CV64/3D (Picasso96), 2 Mb Chip 72 Mb Fast, OS 3.9 1 Gb SCSI HD + 9.5 Gb IDE HD - A500 68000 ECS, 1 Mb Chip 1.5 Mb Slow 2 Mb Fast , OS 3.1 20 Mb IDE-XT + 50 Mb + 115 Mb SCSI HDs - A4000 68040 AGA + CV64/3D (CGfx 3), 2 Mb Chip 16 Mb Fast , OS 3.9 250 Mb IDE + 1 Gb + 512Mb + 4Gb SCSI HDs - A3000 68030/68882 ECS, 2 Mb Chip 16 Mb Fast , OS 3.5 100 Mb + 2 Gb + 4 Gb + 8.6 Gb SCSI HDs - A500 68040(PP&S) ECS , 1 Mb Chip 2 Mb Fast16 8 Mb Fast32 , OS 2.1 165 Mb SCSI HD

What has not been tested at all is operation on PPC accelerator boards (I don't have one). The library itself completely ignores the PPC at present but there is always the possibility that some of the patches made to exec will be incompatible with the PPC software. Note that the PowerUp PPC kernel is already considered 'unsupported' as it is not compatible with Thomas Richter's mmu.library (which will be used more extensively in future versions of Sysmon). Don't blame me or Thomas about this, we're not responsible for the lack of information about the PowerUp kernel internals. On the other hand, any information about the behaviour of this software with the WarpUp PPC kernel installed will be welcome.

This software is intended for experienced users. In particular, there are no GUI tools available yet; so if you are a mouse-maniac user that don't like typing Shell commands or editing ASCII configuration files, I'm afraid that this software is not for you ;-) .

1.3 License

LICENSE

Sysmon is Freely Distributable Copyrighted Software (FreeWare).

It may be freely copied as long as it is kept intact. However, the support commands that do not require sysmon.library may be distributed separately. (see the Other Commands section).

It may not be sold under any guise. I don't want others to make money out of my work. Distribution MUST BE FREE, except for media costs plus a token amount covering only shipping and handling.

Authors of FreeWare may freely use the sysmon.library in their software. Authors of ShareWare may use the library at the cost of giving me a free registration (this includes upgrades as long as the software continues to use the library). Authors of commercial packages MUST obtain a written license agreement from me before selling software using the library.

Inclusion of this archive on a CD-ROM compilation of freely distributable software is allowed provided that the retail price of this CD-ROM is not greater than 15 Euros. Furthermore, a free copy of this CD-ROM must be made available for me at no charge except shipping costs.

Usage and Distribution of Sysmon and sysmon.library is STRICTLY FORBIDDEN to anyone affiliated with MICROSOFT CORPORATION or its subsidiaries and to people working in the design, production or sale of any kind of WAR WEAPONS.

This software is provided "as is" without any warranty, either expressed or implied. By using this software, you accept the entire risk as to its quality and performance.

Especially, as this software patches some private system functions, it can not be guaranteed that it will continue to work with future operating system versions, and it may not work at all on a PowerPC based Amiga.

1.4 Contacting me

CONTACTING THE AUTHOR

I can be currently reached by E-Mail at the following address: Etienne.Vogt@obspm.fr

Updates to Sysmon and my other Amiga products (like vdisk.device) are available from my web page at the address http://mesopk.obspm. and also on Aminet for the latest release versions.

Bug reports should include the complete hardware and software configuration, including any third party software that was running in the background. The Output from 'ShowConfig' and 'ShowSys' may be useful.

If you have any question or suggestion about this software, feel free to send me E-Mail.

1.5 Installation

INSTALLATION

- Copy the contents of the 'libs/' sub-directory to LIBS: (Copy libs/* LIBS: ALL CLONE) Note that the sysmon.library will actually be copied to a sysmon subdirectory in LIBS: to avoid accidental activation of the library on a fully running system, where the patches made by the library initialisation code to already running tasks may not be safe to apply.

- Copy the support commands in the 'c/' sub-directory to C:

- Copy the files in the 's/' sub-directory to S:

- You may also copy the files in the 'autodocs/', 'fd/' and 'include/' subdirectories wherever you like. These are intended for developpers.

- Load S:Startup-Sequence in your favorite editor. Locate the line with 'SetPatch' in it. Add a line just before it with SetTrapVectors . Add a line just after 'SetPatch' with StartSM . Save the modified Startup-Sequence. If you have an accelerator board that is configured via a command in your Startup-Sequence, make sure to put the 'StartSM' command after any command used to configure your CPU or memory. You must also make sure that any program that patches exec.library/ColdReboot() in a "creative" way (anything that may delay the reset or make the function fail if called in supervisor mode), like for example DiskSafe, is started after StartSM or the resets initiated from the sysmon.library may not work correctly. Also set the CACHEREBOOT configuration option if you are using such kind of patches.

- Wait for disk activity to finish, then reboot your Amiga. Sysmon will then be started with the default configuration.

- After bootup, open a shell and try to run the Sysmon monitor program by typing 'sysmon'. If it responds with the 'SYSMON>' prompt, the library is correctly installed. You can exit by typing 'exit' or play with some commands. Refer to the section covering the Sysmon monitor and the Support commands for more information.

1.6 The Configuration File

CONFIGURATION

The configuration file is used to set up various parameters of operation. It is usually located in the S: directory. If you want to put it somewhere else, you have to add the complete path to the file as argument to the StartSM command in the Startup-Sequence. Any line in the file beginning with ';' or '#' is considered to be a comment line and is ignored.

The configuration parameters currently available are :

- LOGFILE : This parameter sets up the name of the file used for the system message logging facility. Currently all messages go into a single file. You should place this file on a hard disk partition with sufficent free space available. The default file name is S:Sysmon.log

- LOGWINDOW : This sets up the window specifications for syslog messages that are serious enough to be displayed in a window. You can also redirect them somewhere else like an AUX: serial terminal or even a file, but you must be aware that this 'file' is not closed after each message like the log file is. The default window specification is CON:20/50/600/80/SysLog/AUTO/CLOSE/WAIT

- FILEPRI : Sets up the higher priority level of messages that will be logged in the log file; low values mean high severity like in UNIX (See the sysmon.h include file for the definitions). The default value is 7 (LOG_DEBUG) which means that all messages will be logged.

- WINPRI : This is the same as the previous parameter but for the window logging. The default value is 4 (LOG_WARN), meaning that messages of class warning or more severe will be logged.

- CONPRI : Sets up the higher priority level for logging to a serial port terminal (console). The logging is done via the smKPrintf() function which is not system friendly (accesses the serial port hardware directly). If you have something else than a terminal attached to your serial port (ex. a modem), you should leave this parameter to its default value of 0 (LOG_EMERG). This means that only system panic messages will be logged; these can not be disabled but are not currently used by the sysmon package (and should never be used by application or user programs).

- LOGBUFFERS : This sets up the number of preallocated buffers to store syslog messages before they can be logged to disk and/or window by the Sysmon.server process. If you have a program that generates lots of messages and you think you are losing some of them, you should increase the number of buffers. The default value is 5.

- STAMPPERIOD : Sets the time period in minutes between time stamping messages written to the logfile at priority level 6 (LOG_INFO). You can disable these messages by specifying a stamp period of 0. Default value is 60 minutes (one message every hour).

- IDLELED : This enables you to monitor the CPU activity using the power LED. If IDLELED is set to YES, the system will dim the power LED when the CPU is idle and brighten it when a task is running. This can however cause interferences with audio output as the power LED also controls the low pass audio filter. Default value is NO.

- TRAPRESET : (V1) Allows to trap keyboard resets and execute a quick system shutdown procedure before the system actually resets. When TRAPRESET is set to YES, the Sysmon.server process will launch a subprocess with the command 'Execute S:Shutdown REBOOT QUICK NOCONF'. This will effectively stop all disk activity before allowing the system to reboot. See the description of the Shutdown script for more details. Note that this feature is not supported by all Amiga models. Most systems with a separate keyboard should work (as long as it is a standard Amiga one). On the other hand, the A500 does not support keyboard reset trapping. The default value for TRAPRESET is NO.

- MOVEVBR : (V1) This will move the exception vector table into fast memory if it is not already there. It won't do anything on a 68000 system since this processor lacks the VBR register. Defaults to NO.

- RAMLIBPATCH : (V1) This will correct (by brute force hacking, yuck !) a bug/misfeature of the ramlib process that abuses the SIGB_SINGLE signal bit for its message port, causing problems when a library init routine has to wait for a semaphore (this can cause hangs with SnoopDOS for example). This patch tries to find ramlib's message port and to force it to use another signal bit, taken from the process' free ones. Defaults to NO.

- OLDMMURESET : (V1) This will make sure that the boot MMU configuration is restored (using mmu.library V42) before a reset generated through the keyboard (if TRAPRESET is also enabled) or software is allowed to occur. In the case of a pending keyboard reset, this requires to call the keyboard.device KBD_RESETHANDLERDONE command from supervisor mode (which is a bit dirty), so this option should only be activated on systems which fail to reset properly otherwise, like some softkicked A3000s. Defaults to NO.

- CACHEREBOOT : (V1) This option will cause the library to call the exec ColdReboot() function via a private vector cached at boot time in order to bypass patches that may interfere with resets. The sysmon.library needs in some cases to be able to reset the system even from supervisor mode or with all tasks frozen. Activate this option in case you are running software that interferes with software generated resets (like for example THOR's DiskSafe). Note that the reset patching software must be started after StartSM for this to work. Defaults to NO.

- MACOSKLUDGE : (V1.17+) This option should only be activated if you want to use MacOS emulators like ShapeShifter or Fusion together with Sysmon. Sysmon normally catches attempts to perform a task switch while the CPU is in supervisor mode (something that is expressly forbidden under AmigaOS) and crashes the system with an AN_smSuperTaskSwitch (C0000007) deadend alert. As MacOS emulators appear to call exec.library/Wait() in supervisor mode, this switch can be used to ignore that bug, hoping it won't corrupt anything. For sysmon.library 1.18 and up, this option also deactivates the supervisor stack pointer validity checks in smExitIntr(). Defaults to NO.

- QUANTUM : (V1.18+) This keyword allows you to set the CPU scheduling quantum, that is, the amount of time in units of vertical blank interrupts that a task will be allowed to run if it doesn't voluntarily yield the CPU or isn't preempted by a higher priority task. When the quantum expires, a rescheduling is forced so that other tasks of the same priority will be allowed to run in a round robin fashion. Note that changing this value has actually few impact on the system multitasking since preemption by regularly active high priority tasks (such as the input.device) often occurs before quantum expiration even at low quantum settings. Defaults to 4 which is the normal Exec setting.

1.7 Support Commands

SUPPORT COMMANDS

These support commands should be installed in your C: directory. They can only be used from the Shell. The commands currently provided are:

- StartSM - SetTrapVectors - ShowSys - Timer - Freeze - UnFreeze - Halt - SysLog - Broadcast - AlertDump - BcMTest - UpTime

1.8 StartSM

StartSM:

This command loads and initializes the sysmon.library and starts the Sysmon.server process. It should be called from your Startup-Sequence right after the SetPatch command.

Usage : StartSM CONF=CONFIGFILE,D=DEBUG/S

CONFIGFILE : Specifies the path to the configuration file described before. It defaults to S:Sysmon.config. DEBUG : Prints some extra diagnostic info to the shell for debugging purpose.

1.9 SetTrapVectors

SetTrapVectors:

This command should be run just before SetPatch in your Startup-Sequence, else it will refuse to install itself. It will modify the default exception vectors to allow proper logging of CPU exceptions from supervisor mode that lead to deadend alerts. The default code in exec jumped directly into the alert function without using the library vector offset.

Usage : SetTrapVectors

1.10 ShowSys

ShowSys:

This command will give a list of all the tasks in the system with their address , name , state , priority , cumulated CPU usage and type .

Usage : ShowSys FULL/S

FULL: This switch causes the command to also display the allocated stack size, signals usage, task flags and total dispatch count for each task.

An example of output on my old 68000 A500 is shown here:

AmigaOS 40.63 at pulcino 05-Nov-95 19:34:37 UpTime : 0-04:45:06 Address Name State Pri CPU [68000] Type 0020F700 DH0 WAIT 10 0-00:00:20.761 Process 0029E200 Workbench (Workbench) WAIT 1 0-00:01:38.599 Cli 4 0021B918 DH2 WAIT 10 0-00:00:37.442 Process 002A7628 Active WAIT 21 0-00:00:31.385 Process 00267930 « ConClip » WAIT 0 0-00:00:00.002 Process 00217138 SH0 WAIT 10 0-00:00:02.803 Process 00223A3C trackdisk.device WAIT 5 0-00:00:25.848 Task 0020703A input.device WAIT 20 0-00:22:01.744 Task 002B9C40 Clock WAIT 0 0-00:04:03.565 Process 00254248 « IPrefs » WAIT 0 0-00:00:03.761 Process 002AFE50 Spliner WAIT 0 0-00:00:03.037 Process 0022C758 ramlib WAIT 0 0-00:00:00.762 Process 00228458 DH3 WAIT 10 0-00:00:02.828 Process 0020B858 scsi.device WAIT 11 0-00:00:15.128 Task 00214C70 DF0 WAIT 10 0-00:00:04.401 Process 00221570 DF1 WAIT 10 0-00:00:03.449 Process 00246570 ErrorLog.daemon (ErrorLogD) WAIT 6 0-00:00:23.148 Cli 0 0026A978 RexxMaster WAIT 4 0-00:00:00.399 Cli 0 00297E78 CON SWDOS 5 0-00:02:08.857 Process 002B1D78 ClickToFront WAIT 21 0-00:00:00.194 Process 0020DA8C trackdisk.device WAIT 5 0-00:00:35.705 Task 00241A88 VD0 WAIT 10 0-00:00:35.972 Process 0020C490 A590 SCSI handler WAIT 12 0-00:00:24.504 Task 002B5890 AppMenuCx WAIT 0 0-00:00:00.198 Process 00285C98 Background CLI (Snap) WAIT 1 0-00:00:13.573 Cli 2 0023A4A0 RAM WAIT 10 0-00:00:02.588 Process 002A49A0 AmigaEyes READY -1 0-00:10:44.476 Process 00223CB0 DH1 WAIT 10 0-00:00:07.265 Process 00296AD0 AmigaShell_3 (showsys) RUN 0 0-00:06:02.147 Cli 3 002339D8 Sysmon.server WAIT 3 0-00:00:00.106 Process 0020B2F8 console.device WAIT 5 0-00:00:00.400 Task

1.11 Task Address

A task's address is actually the address of the Task structure (also known as Task Control Block or TCB) that describes it. These are allocated by exec.library for every task in the system.

Sysmon.library actually manages a second structure for tasks called the TaskInfo structure. It is used to store things like the task CPU usage that are not found in the normal TCB. See the include file sysmon.h for a description of this structure.

1.12 Task Name

This is the string pointed to by the ln_Name field of the Task structure. If this pointer is NULL (shouldn't happen, but...) "<< No Name >>" will be displayed.

If the task is a Cli process with a command currently loaded, the name of this command (without path) will be added inside parenthesis after the task name.

The complete lenght of this field (task name + command name) is limited to 32 characters.

1.13 Task States

A task can be in several scheduling states. Sysmon.library adds more states to those that were initially available in exec. Also some special cases of the WAIT state will be shown as pseudostates.

The states that were initially defined in exec are :

- INVLD (TS_INVALID) Invalid state. You shouldn't see this one.

- ADDED (TS_ADDED) A task that was just added by AddTask() but not yet linked into the TaskReady queue. This is a very transient state that you shouldn't normally see.

- RUN (TS_RUN) The task that the CPU is currently running. This will always be the task executing the task display command.

- READY (TS_READY} A task that is ready to use the CPU but that is not currently running. These tasks are linked into the TaskReady queue, sorted by priority.

- WAIT (TS_WAIT) A task that is waiting for a signal in a mask to become set, having called the exec Wait() function. When any of the signals waited for becomes set, the task will return to the READY state. WAITing tasks are linked into the TaskWait list. Some special cases are displayed as pseudo-states: - STOP A task that is waiting for no signals at all, being stuck in a Wait(0) call, such as crashed tasks that have been suspended. These tasks will never return from the Wait(). - SWABO A task Waiting for the SIGB_ABORT Signal that is not used by anything as far as I know. - SWCHI A task waiting for the SIGB_CHILD signal. Again, what is this used for ? - SWSIN A task waiting for the SIGB_SINGLE signal, normally used when waiting for a semaphore (and also abused by ramlib unless patched). May also be used for blitter operations. - SWINT A task waiting for the SIGB_INTUITION signal, but is this really used at all ? - SWNET A task waiting for the SIGB_NET signal, used in particular by the Envoy networking software. - SWDOS A task waiting for the SIGB_DOS signal, used for all dos.library I/O. - SWBRC A task waiting for the SIGBREAKB_CTRL_C signal, used (guess what) by the CTRL-C key combination. - SWBRD Same but for CTRL-D - SWBRE Same but for CTRL-E - SWBRF Same but for CTRL-F

- EXCPT (TS_EXCEPT) A task that is ready to be dispatched into an exception routine. Note that this state was never used at all by exec as far as I know despite being defined in the include files (exec code uses the TB_EXCEPT bit in tc_Flags). It is now made to good use by the new exception routines in sysmon.library to indicate a task that is really ready to enter exception code (it is not already running exception code and exception delivery is not disabled, the TB_EXCEPT bit in tc_Flags being used to indicate that an exception is pending). These tasks are linked into the TaskReady queue.

- REMVD (TS_REMOVED) A task that is in the process of being removed by RemTask() but that has not been completely deleted yet. This is again a very transient state.

The following states are new in sysmon.library :

- FROZN (TS_FROZEN) A task that has been frozen (suspended) by the smFreeze() function. Such tasks are no longer dispatched by exec until unfrozen (resumed) by smUnFreeze(). Exception delivery is also disabled for frozen tasks. A pending exception will be delivered when the task is unfrozen. Frozen tasks are linked into a private list.

- HIBER (TS_HIBERNATE) A task that has put itself into hibernation (sleep) by calling smHibernate(). It will stay in this state until woken up by an smWakeUp() call. In contrary to frozen tasks, hibernating ones will be temporarily awoken if they receive an exception signal. Hibernating tasks are linked into the TaskWait list.

- PGFWT (TS_PAGEFLTWAIT) A task that is waiting for page fault processing by the Swapper daemon. This is not used currently but reserved for a future virtual memory manager module.

- WAITA (TS_WAITAND) A task that is waiting for two or more signals to be simultaneously set, having called smWaitAnd(). The task will only return to the READY state when all signals waited for are set. These tasks are linked into the TaskWait list.

- TRAP (TS_TRAP) A task that has been suspended by exception trap code. This is reserved for future use.

- FREWT (TS_FREEWAIT) A task that is waiting for free public memory to be released. This is reserved for use by a future virtual memory manager module.

- ????? (undefined) Undefined state. You shouldn't see this one.

1.14 Task Priority

The priority determines how the CPU will be shared between tasks. The rule is that the task with the highest priority will always run until it voluntarily gives up the CPU (entering a wait state) or is forcefully preempted when a higher priority task becomes ready (note however that tasks can Forbid() preemption with a well known exec.library call).

If there is more than one task ready to run at the same priority, exec will share the CPU between them in a round robin fashion, giving the CPU to each task for a predefined quantum of time (usually 4 vertical blank periods, 200 milliseconds on a 50 Hz system).

The priority value is a signed byte, thus ranging from -128 to 127. Normal user processes run at priority 0.

1.15 Task cumulated CPU usage

The cumulated CPU usage of each task is maintained by sysmon.library. It is internally stored as a 64 bit EClockVal. It is converted for display to a d-hh:mm:ss.mmm format where d is the number of CPU days used (you will see this if you keep your machine running to crack RC5 keys ;-)), hh the number of hours, mm for minutes, ss for seconds and mmm for milliseconds.

Note that CPU usage monitoring starts when the library is initialized so the CPU already used by the tasks that were started before the library was loaded can't be counted.

1.16 Task Types

There are actually different types of tasks in an AmigaOS system. This comes from the fact that a very important part of the OS, the dos.library, was initially ported from another operating system (Metacomco's TripOS) and thus didn't fit perfectly within the task model of exec. Any task that needs access to the dos.library needs an additionnal environment known as a Process structure, that is an extension of the basic Task structure. Only lowlevel system tasks such as device drivers don't actually need access to the DOS, so most tasks in the system are actually processes. Again, there are two types of processes depending on weather they have a CLI (Command Line Interface) structure attached to them or not. This CLI structure provides the additionnal environment needed for the Amiga Shell.

In summary, basic tasks will be shown with the type 'Task', processes without a CLI attached as 'Process' and CLI processes as 'Cli' followed by the CLI process number (as used by the Shell). A process number of 0 indicates that the process has a CLI structure attached to it, but that it is not actually linked into the DOS CLI tables (such a process was created with CreateNewProc() using the NP_CLI tag).

1.17 Timer

Timer:

This command runs another command as a subprocess and times the command execution. It then reports the elapsed time and the CPU time used by the command. It is similar to the UNIX 'time' command (this name was already taken by the Time preference editor).

Usage : Timer COMMAND/F/A

COMMAND : The command that will be executed and timed.

1.18 Freeze

Freeze:

This command will suspend a given task by putting it in the FROZEN state , using the sysmon.library function smFreeze(). This can be useful to stop a looping task that is eating away all the CPU time available. Be careful though not to freeze system tasks or the whole machine may hang.

The frozen task can be resumed by the UnFreeze command.

Usage : Freeze TASK,ADDR=ADDRESS/K

TASK: The name of the task to be frozen. If it is not unique, the first task found in the system lists will be frozen. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the task as displayed by ShowSys .

1.19 UnFreeze

UnFreeze:

This command will bring a previously frozen task back to life using the sysmon.library function smUnFreeze().

Usage : UnFreeze TASK,ADDR=ADDRESS/K

TASK: The name of the task to be unfrozen. If it is not unique, the first frozen task found will be unfrozen. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the task as displayed by ShowSys .

1.20 Halt

Halt:

This command will halt the system by displaying a guru-like deadend alert via the sysmon.library function smHalt(). It is used by the shutdown script and is not intended to be called directly as it will crash the system without caring about ongoing disk activity (that's what Shutdown is for).

Usage : Halt REBOOT/S,REKICK/S

REBOOT : This switch bypasses the alert display and causes the system to reboot immediately. REKICK : This switch forces MMU-softkicked machines like the early A3000s to reload the kickstart file when rebooting. It requires the mmu.library from Thomas Richter to be installed. It will also clear the ExecBase capture and KickTag vectors, so the ROM update modules from AmigaOS 3.5/3.9 will be reloaded.

1.21 SysLog

SysLog:

This command can be used to generate syslog messages from the Shell using the smVSysLog() function in sysmon.library. The facility code is currently fixed to LOG_USER.

Usage : SysLog MESSAGE/A,LEVEL/K/N,NOHEAD/S,NOWIN/S,NOFILE/S

MESSAGE : The message text body which must be inserted between double quotes if it contains spaces. The message length is limited to SM_MAXLOGCHARS bytes (currently 256). Extra characters will be truncated. LEVEL : The priority level of the message in numeric form (see sysmon.h) Do not use the LOG_EMERG (0) and LOG_ALERT (1) values without good reasons. Defaults to LOG_DEBUG. NOHEAD : This switch suppresses the header that is normally prepended to each message. This allows you to split a single long message in smaller parts. NOWIN : Do not log this message to the log window regardless of priority thresholds. NOFILE : Do not log this message to the log file regardless of priority thresholds.

1.22 Broadcast

Broadcast:

This command allows you to send a systemwide broadcast message via the smSendBroadcastMsg() function of sysmon.library to all ports that have been registered via the smAddBroadcastPort() function. It is used by the Shutdown script to send shutdown notifications.

Usage : Broadcast MESSAGE/A,COUNTDOWN/K/N,TIMEOUT/K/N,CANCEL/S,DEBUG/S, NORMAL/S,URGENT/S,SHUTDOW

MESSAGE: The text of the broadcast message. COUNTDOWN: Countdown to the announced event in seconds. Defaults to 0. TIMEOUT: Message retrieval timeout by clients in ticks (1/50th of a second). Defaults to 50 (1 sec). CANCEL: Sets the flag indicating that this message cancels the previous one. DEBUG: Send the message at the DEBUG priority level. NORMAL: Send the message at the NORMAL priority level. This is the default. URGENT: Send the message at the URGENT priority level. SHUTDOWN: Send the message at the SHUTDOWN priority level. Reserved for the shutdown script. UNMOUNT: Send the message at the UNMOUNT priority level. Reserved for the shutdown script. HALT: Send the message at the HALT priority level. Reserved for the shutdown script.

1.23 AlertDump

AlertDump:

This command will display the extended alert information stored by the new exec Alert() function installed by sysmon.library. It will in particular display the name of the task running when the alert was posted, a dump of the CPU registers as seen from the Alert() function and it will also translate the Guru Meditation number to something more human-readable (The command knows most alert codes). Note that to capture crashes originating from CPU traps while in supervisor mode, the SetTrapVectors command must be run in your Startup-Sequence just before SetPatch. (exec calls its Alert code directly without going through the library jump vector :-( ).

An automatic logging of this alert data will be provided in the future.

Usage : AlertDump CLEAR/S

CLEAR: Invalidates the alert buffer after display.

1.24 BcMTest

BcMTest:

This is a test client for the message broadcast function. It should be run in a shell window where it will print the broadcast messages received.

Usage : BcMTest PRI=PRIORITY/K/N,DEBUG/S

PRIORITY: The priority of the broadcast port created. DEBUG: Prints additional debug information for each message received.

1.25 UpTime

UpTime:

This is a fairly complete Unix-like uptime command, based on the example provided with the original Executive-based SysInfo.library by Petri Nordlund, that has been modified for use with Sysmon. This command is public domain and its source is included in this archive.

It supports multiuser.library to show the number of users currently logged in. If it's not available, it will assume that the system is single user.

It does not take any arguments.

1.26 Other Commands

OTHER COMMANDS

The following Shell commands do not depend on sysmon.library and may be distributed separately from the Sysmon package. I wrote them for my own usage and decided to distribute them with sysmon because they may be useful to the Amiga community. The UnMount command is also used by the shutdown script.

- RunBackground - UnMount - Request - ValidateWait

1.27 RunBackground

RunBackground:

This command starts a command as a detached process. Unlike the standard 'run' command, it automatically redirects the standard input and output to NIL: so that the shell it was launched from can be closed and it also allows you to specify additional options such as the new process priority and stack size.

Usage : RunBackground STACK=STACKSIZE/K/N,PRI=PRIORITY/K/N,DELAY/K/N,NOREQ/S, COMMAND/F/A

STACKSIZE : The size of the stack to allocate for the new process. Defaults to the current shell stack size. PRIORITY : The priority to give to the new process. Defaults to the current process priority. DELAY : Optional delay after the new process startup and before the command returns. This can be used to avoid concurrent accesses to a disk. NOREQ : This switch will disable DOS requesters for the newly created process. Unfortunately, software error requesters are also disabled, so a CPU trap will cause an immediate guru reboot. COMMAND : The command to be executed.

1.28 UnMount

UnMount:

This command will try to unmount a DOS device by sending it an ACTION_DIE packet and optionally an ACTION_INHIBIT packet if the ACTION_DIE fails. It is used by the shutdown script to stop filesystem accesses before halting the system. Note that very few filesystems or handlers currently implement the ACTION_DIE packet (known exceptions being the CrossDosFileSystem, the OS3.5 CacheCDFS and the Envoy3 EFS client).

Usage : UnMount DEVICE,INHIBIT/S,RETRY/K/N,ALL/S,FREENODE/S,QUIET/S

DEVICE : The name of the DOS device to unmount. INHIBIT : This switch causes the command to send an ACTION_INHIBIT packet if the ACTION_DIE fails. ACTION_INHIBIT causes the filesystem to appear as 'BUSY' on the workbench screen, like during a format or diskcopy and effectively stops filesystem activity. RETRY : The number of retries for the ACTION_INHIBIT if the first one is rejected because the filesystem is busy. The retry period is 1 second and the default number of retries is 10. ALL : Finds all mounted filesystems and tries to unmount them (except RAM: and ENV: if mounted as a separate handler). Used by shutdown script. FREENODE: This switch causes the command to remove and free the device node after the handler has been stopped. It will also attempt to free things attached to the device node like a file system startup message, using some heuristics taken from Ralph Babel's Guru Book. Note that this can be dangerous as some handlers may use some fields in very 'creative' ways. The 'Assign device: DISMOUNT' command is safer as it will only free the device node itself. QUIET : Suppresses the display of informational messages. Errors are still displayed on the screen.

1.29 Request

Request:

This command enables/disables system requesters in the current shell. This can be useful for remote shells or in the User-Startup if you make assigns to external disks that may not be always connected.

Usage : Request OFF/S,ON/S,WB/S

OFF : Disables system requesters in the current shell. Unfortunately, software failure requesters are also disabled causing an immediate guru reboot in the event of a CPU trap. ON : Enables system requesters on the current screen. For a remote shell, enables system requesters on the default public screen. WB : Enables system requesters on the default public screen, usually the Workbench screen. This is the default setting at startup.

1.30 ValidateWait

ValidateWait:

This command waits until the specified volume is validated. If it does not become validated within a specified period, it will time out and return an error condition to the Shell. It may be a good idea to put a 'ValidateWait SYS:' command at the beginning of your Startup-Sequence.

Usage : ValidateWait DRIVE/A,TIMEOUT/K/N

DEVICE : The device (or assign) to check for validation. TIMEOUT: The amount of time in seconds before timing-out. Defaults to 300 seconds (5 minutes).

1.31 Shutdown Script

THE SHUTDOWN SCRIPT

The shutdown script should be placed in your S: directory. It enables you to perform a safe power down of your system. This is particularly useful if your machine is connected to a network or runs a BBS at it will make sure that your disks are no longer accessed by something when you power down your system. You should not edit this script as it may change in a future version of the Sysmon package. Instead, you should create a User-Shutdown script in S: where you can put commands to customize the shutdown sequence.

The sequence of events performed by this script is as follows :

- First ask the user for confirmation, unless the NOCONF switch was specified on the command line. If you enter 'y', the shutdown continues, else it is aborted. - Set the current shell priority to 2 and send an initial broadcast message (using the Broadcast command) announcing the shutdown. - If you specified a countdown, wait for the specified amount of time and send broadcast messages if necessary at 5, 3, 2 and 1 minutes before shutdown. You can cancel the shutdown during this time with a CTRL-C. - When countdown is zero, send another shutdown broadcast message as well as a syslog message at level 4 (LOG_WARN). - Execute the S:User-Shutdown script if it exists. You can put commands in this script to stop commodities or network protocol stacks for example. You can also create a Final-Shutdown script in RAM: (via echo or copy commands) that will be executed later. All commands to be used in this Final-Shutdown script must be copied to RAM: - Copy all further needed commands to RAM:, reset the search path to RAM: - Send an UNMOUNT broadcast message then find and unmount all filesystems except RAM:, using UnMount ALL INHIBIT. - Execute the RAM:Final-Shutdown script if created by User-Shutdown. - Send the final HALT broadcast message and halt the system via the Halt command. If the REBOOT and/or REKICK options were given on the command line, they are passed to the Halt command. The REBOOT option will cause the system to reboot immediately. If it is not specified, the Halt command will freeze all tasks except the current one and display a guru like deadend alert saying 'System Shutdown Complete'. You can then safely turn the power off or reboot by pressing the left mouse button.

Usage : Shutdown COUNTDOWN,REBOOT/S,REKICK/S,NOCONF/S,QUICK/S

COUNTDOWN: The number of minutes before shutting down. Defaults to 0. the maximum value allowed is 10 minutes. REBOOT : This switch is passed to the 'Halt' command at the end of the script. It bypasses the alert display and causes the system to reboot. REKICK : This will force MMU-Softkicked machines like early A3000s to reload the kickstart file when rebooting. NOCONF : Skip the confirmation question. QUICK : If this switch is set, the User-Shutdown script will be skipped and the procedure will execute as fast as possible. This is to be used in case of a trapped keyboard reset where the shutdown must execute in less than 10 seconds.

1.32 Sysmon Monitor

THE SYSMON MONITOR

The sysmon monitor is an interactive shell based program that allows you to display the system lists and variables and interact with them (not fully implemented yet). This program is still under development and more commands will be implemented in future releases.

When you type 'sysmon' on the command line, you will get the 'SYSMON>' prompt back that tells you that the monitor is waiting for commands. You can then type sysmon commands (not to be confused with the shell-based sysmon support commands) or exit with the 'EXIT' or 'QUIT' commands.

You can also specify a command directly on the command line. In this case, Sysmon will execute the command and exit immediately thereafter.

All sysmon commands can be abreviated to the minimum number of characters required to uniquely identify the command. The required characters are shown in capital letters in the description below. This is generally valid only for the first two command words as the additionnal arguments are parsed with ReadArgs() and thus need to match the template provided.

Currently implemented commands are :

- EXit - QUIT - SHow - SET - REMove

1.33 Exit/Quit command

This command will exit the sysmon monitor. An end of file (CTRL-\) at the 'SYSMON>' prompt will do the same.

1.34 Show command

This command is used to display information in various system lists as well as details about a particular system object. The type of object to display is given as a second command word after SHOW.

The following objects are currently implemented :

- SHow TAsk - SHow LIBrary - SHow DEVice - SHow RESOurce - SHow RESIdent - SHow POrt - SHow SEMaphore - SHow MEMory - SHow INTerrupt - SHow FOnt - SHow MEMHandler - SHow INPuthandler - SHow RESEthandler - SHow BOard - SHow LOad - SHow KIcktags - SHow BRoadcast - SHow VECtors - SHow SYSBase - SHow FILESYstems

1.35 Show Task command

Without further arguments, this command will display all the tasks in the system, in a manner similar to the ShowSys support command. If a task name or address is given, details about this task will be displayed. This includes a dump of the Task structure, of its Process extension (if the task is an AmigaDOS process), of the TaskInfo structure and of the CommandLineInterface structure for CLI processes.

Usage: SHow TAsk NAME,ADDR=ADDRESS/K

NAME: The name of the task to be displayed. If it is not unique, the first task found will be displayed. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the task. A value of 0 refers to the current task.

1.36 Show Library command

This command will display all shared libraries currently loaded in memory with version number, open count, negative and positive sizes of the library node and node priority.

1.37 Show Device command

This command will display all devices currently loaded in memory with version number, open count, negative and positive sizes of the device node and node priority.

1.38 Show Resource command

This command will display all resources currently known by exec with version number and node priority. Note that the version number may be meaningless for some resources.

1.39 Show Resident command

This command will display the resident modules (also called RomTags) known to exec. This list is built at boot time by scanning the ROM areas and adding the modules linked into the KickTag array. The address of the RomTag is displayed as well as the module name, version, start priority, flags and node type.

1.40 Show KickTag command

This command will display the resident modules (also called RomTags) added through the KickTag array mechanism. This provides a way to replace KickStart modules or to add new ones to the system. It is used in particular by the OS 3.5 to install the new hard disk drivers and filesystem. The sysmon.library also uses it to add a new alert hook module providing the new alert display also to deferred deadend alerts.

The address of each KickTag found is displayed as well as the module name, version, start priority, flags and node type. If the KickCheckSum value is incorrect, a warning message will be printed (an incorrect checksum will cause the KickTag array to be ignored on the next reboot). You should also note that all modules listed here will not necessarily appear in the list displayed by the Show Resident command if they have been added since the last system restart and thus have not been activated yet.

1.41 Show Port command

This command will display the public message ports currently known by exec with port type, priority and also signal bit and task if appropriate.

1.42 Show Semaphore command

This command will display the public signal semaphores currently known by exec with node priority, nesting count, queue count and current owner task.

1.43 Show Memory command

Without further arguments, this command will display the memory headers currently known by exec with their priority, attributes, lower and upper boundaries and free byte count.

If a memory header name or address is given, detailed information about this memory header will be shown. The free list will also be checked and the largest and smallest block size, the number of free chunks as well as the number of chunks smaller than 64 bytes will be displayed, providing information about memory fragmentation in this memory region.

Note that if a wild chunk link pointer (pointing outside the memory region boundaries or to an address lower than or equal to the current chunk one) is encountered during the free list walk, the system will crash with guru number 0x81000011 (you're already lucky to not have crashed earlier) and if the total of the free chunk sizes does not match mh_Free, a recoverable alert 0x0100000c will be posted.

Usage: SHow MEMory NAME,ADDR=ADDRESS/K,FRAGS/S

NAME: The name of the memory region to be displayed. If it is not unique, the region with the highest priority will be displayed. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the memory header. FRAGS: This option selects the output of detailed fragmentation statistics. The number of free chunks by powers of two intervals is displayed as well as the search length for allocating a block with a size given by the lower bound of the interval.

1.44 Show Interrupt command

This command will display the interrupt nodes currently known to exec for each of the 16 IntVectors defined (corresponding to the 14 interrupts in the Paula chip plus the special (and normally unused) INTEN and NMI vectors).

For each IntVector, the corresponding IPL (CPU Interrupt Priority Level), type of vector (may be handler, server, softint, free, unused or unknown type), pseudo priority, code and data addresses are displayed. If the vector is a handler, the currently attached node is displayed below with its address, name, code and data addresses (the priority is meaningless for interrupt handlers). In the case of an interrupt server chain, all attached interrupt servers are displayed, sorted by priority.

1.45 Show Font command

This command will display the Text Fonts currently known by the graphics.library (Fonts that have been loaded into memory). For each font, the address, name, YSize, XSize (only meaningful for fixed width fonts), open count, flags, style, low and high characters and node priority are displayed.

1.46 Show Memhandler command

This command will display the low memory handlers currently registered with exec (only available for AmigaOS 3.0 and higher). The address, priority, name, code and data pointers are displayed for each memhandler.

1.47 Show Inputhandler command

This command displays the input handlers currently registered with the input.device. As this list is not public, a dummy handler is installed (the "Sysmon Input Probe") and its link pointers are walked down to get to the list header. The list is then displayed, showing the address, name, priority, code and data pointers for each handler installed. Finally, the probe handler is removed.

1.48 Show Resethandler command

This command displays the reset handlers currently registered with the keyboard.device. As this list is not public, a dummy handler is installed (the "Sysmon Reset Probe") and its link pointers are walked down to get to the list header. The list is then displayed, showing the address, name, priority, code and data pointers for each handler installed. Finally, the probe handler is removed.

1.49 Show Board command

This command displays the expansion boards installed. The board address, size, type manufacturer ID and product ID will be displayed. If Richard Körber's identify.library (available from Aminet) is installed, it will be used to get the names corresponding to the manufacturer and product IDs.

1.50 Show Load command

This command periodically displays the top CPU eating tasks and total CPU load. Type CTRL-C to interrupt the display and return to the Sysmon prompt.

Usage: SHow LOad UP=UPDATE/K/N,MAX=MAXDISP/K/N,THR=THRESHOLD/K/N

UPDATE : Interval in seconds between updates of the display. Defaults to 5 seconds. MAXDISP : Maximum number of tasks to be displayed, sorted by CPU usage during the update interval. Tasks that have used less than 1 millisecond of CPU during the update interval will not be shown. Defaults to 16. THRESHOLD: Lowest task priority to be taken into account. Tasks with a priority lower than the specified threshold will be ignored. Defaults to -128, meaning that all tasks will be counted.

An example display from a 'quiet' 68000 system is shown below :

UpTime : 0 01:03:31 Update : 00:00:05.010 CPU Load : 20.0 % DispCnt : 165143 IdleCnt : 170715 DispUpd : 120 IdleUpd : 256 Address Name Load CPUTime Dispatch 0029B3D0 AmigaShell_3 (sysmon) 7.7 % 00:00:00.385 15 0020703A input.device 4.6 % 00:00:00.232 42 0029C778 CON 3.7 % 00:00:00.184 15 002B8E90 AmigaEyes 1.7 % 00:00:00.085 17 002BCFC8 Clock 1.3 % 00:00:00.064 4 002A2B58 Workbench (Workbench) 0.6 % 00:00:00.028 10 00218834 trackdisk.device 0.2 % 00:00:00.010 9 002253AC trackdisk.device 0.0 % 00:00:00.003 2

1.51 Show Broadcast command

This command will show the broadcast ports that have been registered with sysmon.library via the smAddBroadcastPort() function to receive broadcast messages sent with the smSendBroadcastMsg() function. The display is similar to that of the Show Port command.

1.52 Show Vectors command

This command will display the contents of the CPU vector table (for interrupts and exceptions) pointed to by the VBR register. This is mainly provided for system debugging as these vectors should normally not be touched in any way. Adding the FULL switch will also show reserved vectors.

1.53 Show Sysbase command

This command will display the internal system variables in the ExecBase and SysmonBase structures. This is for debugging only, as these variables should normally not be accessed in any way, except when explicitely allowed in the system autodocs.

The validity of checksum fields in ExecBase will also be checked and the relevant entries will be marked as Valid or Not Valid. Note that the ExecBase->LowMemChkSum field will always be marked as invalid as it is currently not used by the system.

1.54 Show Filesystems command

This command displays the filesystems that have been registered into the FileSystem.resource. Note that most filesystems don't provide any useful name into the filesystem node name so they are better identified via the DOSType value.

1.55 Set command

This command is used to modify or act upon objects in the system lists. The type of object is given as a second command word after SET, the object name/address and new attributes are given as options in standard ReadArgs() format.

You should know what you do when using these commands, else you may very easily crash the system or cause other unpredictable results.

Currently implemented objects are :

- SET TAsk - SET LIBrary - SET DEVice - SET RESOurce - SET POrt - SET SEMaphore - SET MEMory - SET FOnt - SET MEMHandler

1.56 Set Task command

This command is used to act upon a task, such as modifying its scheduling priority or sending signals.

Usage : SET TAsk NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N,SIG=SIGNAL/K,FREEZE/S, UNFREEZE/S,WAKEUP/S

NAME: The name of the task to be modified. If it is not unique, the first task found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the task. A value of 0 refers to the current task. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the task with the exec function SetTaskPri(). SIGNAL: Specify an hexadecimal mask of signals to be sent to the task through the exec Signal() function. For example, 00001000 will send a SIGBREAKF_CTRL_C signal to the task. FREEZE: Attempt to freeze the task with the sysmon.library function smFreeze(). Use this with caution when dealing with system tasks. UNFREEZE: Resume a frozen task through the sysmon.library function smUnFreeze(). WAKEUP: Send a wakeup request to a task through the sysmon.library function smWakeUp().

1.57 Set Library command

This command is used to act upon a shared library node.

Usage : SET LIBrary NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the library to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the library node, which will be properly requeued into the list.

1.58 Set Device command

This command is used to act upon a device node.

Usage : SET DEVice NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the device to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the device node, which will be properly requeued into the list.

1.59 Set Resource command

This command is used to act upon a resource node.

Usage : SET RESOurce NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the resource to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the resource node, which will be properly requeued into the list.

1.60 Set Port command

This command is used to act upon a public message port.

Usage : SET POrt NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the port to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the port. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the port node, which will be properly requeued into the list.

1.61 Set Semaphore command

This command is used to act upon a public signal semaphore.

Usage : SET SEMaphore NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the semaphore to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the semaphore node, which will be properly requeued into the list.

1.62 Set Memory command

This command is used to act upon a systemwide memory header.

Usage : SET MEMory NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N,ATTR=ATTRIBUTES/K

NAME: The name of the memory header to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the memory node, which will be properly requeued into the list. ATTRIBUTES: Set new attribute flags (as an hexadecimal 16 bit mask) for the memory region. USE THIS WITH EXTREME CAUTION !!!

1.63 Set Font command

This command is used to act upon a TextFont node.

Usage : SET FOnt NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the font to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the font node, which will be properly requeued into the list.

1.64 Set Memhandler command

This command is used to act upon a low memory handler node.

Usage : SET MEMHandler NAME,ADDR=ADDRESS/K,PRI=PRIORITY/K/N

NAME: The name of the memhandler to be modified. If it is not unique, the first node found will be selected. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. PRIORITY: Specify a new priority (-128 to 127) to be attributed to the memhandler node, which will be properly requeued into the list.

1.65 Remove command

This command is used to remove an object from the system lists. The type of object is given as a second command word after REMove, the object name/address are given as options in standard ReadArgs() format.

You should know what you do when using these commands, else you may very easily crash the system or cause other unpredictable results.

Currently implemented objects are :

- REMove LIBrary - REMove DEVice - REMove FOnt

1.66 Remove Library command

This command is used to remove a shared library from the system via the exec function RemLibrary().

Usage : REMove LIBrary NAME,ADDR=ADDRESS/K,FORCE/S

NAME: The name of the library to be removed. If it is not unique, the first node found will be removed. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. FORCE: Attempt to remove the library even if its opencount is non zero. Can be used to force a library to flush internal resources.

1.67 Remove Device command

This command is used to remove an exec device from the system via the function RemDevice().

Usage : REMove DEVice NAME,ADDR=ADDRESS/K,FORCE/S

NAME: The name of the device to be removed. If it is not unique, the first node found will be removed. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. FORCE: Attempt to remove the device even if its opencount is non zero. Can be used to force a device to flush internal resources.

1.68 Remove Font command

This command is used to remove a text font from the system via the graphics.library function RemFont().

Usage : REMove FOnt NAME,ADDR=ADDRESS/K,FORCE/S

NAME: The name of the font to be removed. If it is not unique, the first node found will be removed. ADDRESS: You can alternatively to the name, specify the hexadecimal address of the node. FORCE: Attempt to remove the font even if its access count is non zero and clears the TE0F_NOREMFONT flag in the TextFontExtension structure before attempting to remove the font.

1.69 Contributed Software

CONTRIB

This directory contains software written for Sysmon by other authors.

LastGuruLog.rexx: This ARexx script is contributed by Christopher 'WatchDog' Elliott. It allows extended logging of Alert data into the Sysmon logfile using the AlertDump and SysLog commands. The original script also used a custom Beep command which I replaced by an inline beep character ('07'X in ARexx). It is intended to be used from your User-Startup script and takes one argument, QUIET, to prevent any output to the console.

1.70 Programming with Sysmon

INFORMATION FOR DEVELOPPERS

You can use the sysmon.library in your own programs. See the License for legal details. The library contains public functions that can be used to walk through the system task tables in an elegant way (more than the hacky peeking into ExecBase anyway), use the newly defined scheduling states, send syslog or broadcast messages and more. You will find a detailed description of the available functions in the sysmon_lib.doc autodoc file and description of the data structures in the include files.

Sources for some of the support commands as well as for the Executive compatible SysInfo.library are provided as examples. For the functions provided by the SysInfo.library, please refer to the developper material included in the Executive archive. Starting with the 1.16 release of Sysmon, the SysInfo.library now includes Load Averages, as well as systemwide CPU Usage (total, recent and last second), task switches (total and last second, both voluntary and involuntary). For a given task, total CPU usage

and task switches (voluntary and involuntary) can be reported. Notification may be added in a future release, but please don't ask for nice values support.

Information is also available about the system patches installed by the sysmon.library and about Known incompatibilities or bugs .

I recommend to use the SysInfo.library to get information about CPU usage, rather than peeking into the TaskInfo structures. Also remember that SysmonBase is PRIVATE, peeking here is not supported (except for some of the support commands ;-)) as everything down there is susceptible to change in every new release.

I would like to get some reports on how the new task signal exception mechanism works. It seems that the old one was so badly broken that almost nobody used it at all. If you have some software that makes use of task signal exceptions, please do some tests and report back to me.

1.71 System patches installed

Patches to exec.library :

- Switch() : this private task switching entry is completely replaced. - Dispatch() : also replaced. - AddTask() : Patched to allocate a TaskInfo structure for the new task and link it into the hash table. - RemTask() : Replaced by a new version that properly deallocates the Task and TaskInfo structures. - FindTask() : Replaced by a new version that uses smFindTaskInfo() and thus doesn't need to disable interrupts anymore. - Alert() : Replaced by a new version that saves the alert information to a reset resistant buffer, displays a different alert text based on the general error field of the guru number and brings back the good old 'Guru Meditation' message. This new version also makes use of the mmu.library, if available, to properly store the deferred alert information in the zero page even if it is remapped or marked as invalid without causing muForce hits. A recoverable alert occuring while in supervisor mode is automatically promoted to a deadend one and the system is properly reset via ColdReboot() on a deadend alert. - ExitIntr() : This private entry is replaced. - Schedule() : This private scheduler entry is replaced. - Exception() : This private exception dispatch entry is completely replaced. In particular, the state of the task before the exception is now properly saved and restored and exception nesting is no longer allowed, even for different signals. Instead, the exception routine will be entered again after the first one completes. It is also possible to terminate the task from the exception routine if appropriate precautions are taken. - SetTaskPri() : Replaced by a version that properly handles tasks in the TS_EXCEPT state. An invalid task pointer (checked via smGetTaskInfo()) will also cause a recoverable alert (AN_smNoTaskInfo) - SetExcept(), SetSignal(), Signal() : Replaced by new versions that handle the new exception mechanism and the new scheduling states. Passing an invalid task pointer to Signal() will also cause an AN_smNoTaskInfo alert. - Wait() : Replaced by a new version that will generate a specific alert (AN_smSuperTaskSwitch) when it is called from supervisor mode. - SuperState()/UserState() : Replaced by new versions which inform other parts of the library that the system stack is not valid while SuperState() is in effect. UserState() will also crash with an AN_StackProbe deadend alert if the system stack pointer passed back is not valid. - ExecBase->TaskExitCode : Replaced by a version that properly calls the new RemTask() routine - ExecBase->TaskExceptCode : Replaced by a version that generates a specific recoverable alert (AN_UnInitExcpt) on an uninitialised exception instead of some random crash. - ExecBase->TaskTrapCode : Replaced by a version that properly calls Alert() through the exec jump table.

Additionaly, sysmon.library installs a vertical blank interrupt server that is used to probe the TaskReady queue every second and gather statistics that are used by SysInfo.library to compute load averages.

1.72 Exception processing

Sysmon Task Exception Processing

The sysmon.library provides a new task signal exception mechanism to replace the insufficient and buggy one provided by Exec. A task exception is some sort of private "interrupt" routine for a task, that executes in user mode, sharing the task's context. They are triggered by the reception of specified signals. They should not be confused with CPU exceptions, called Traps in the Exec jargon, that execute in supervisor mode.

The new implementation is mostly compatible to the original Exec one, with the notable exception that no nesting is allowed, even for different signals. Since task exceptions were actually rarely used, I don't expect this to be a problem. Also, sysmon.library provides functions to prevent exception delivery, the smDisallowExcept()/smAllowExcept() pair. Unlike exception signal deactivation via SetExcept(), these provide nesting like Forbid()/Permit() and allow easier implementation of exception safe critical sections.

Like under the old Exec mechanism, you should first set the address of your exception routine in the tc_ExceptCode field of the task structure. By default, this field points to a routine that will display a recoverable alert to warn about uninitialised exceptions ($0100000a AN_UnInitExcpt) instead of some random crash under the original Exec setup (which jumped into a CPU exception handling routine expecting supervisor mode in this case). Setting tc_ExceptCode to NULL will cause the exception to be ignored (the exception pending flag is still properly cleared). tc_ExceptData should be set to hold any data your exception code may require. It will be loaded into register A1. When the exception routine is properly initialised, you use the Exec function SetExcept() to select which signals should trigger an exception. If one of the selected signals is already set, an exception will be triggered immediately, so it is important to first setup the exception routine properly.

When an exception signal becomes set (or a set signal becomes an exception signal), the system will first set the exception pending flag TB_EXCEPT in the tc_Flags field of the task structure. Then, it checks if exception delivery has been disabled via smDisallowExcept(), if an exception routine is already running (TIB_INEXCEPT flag set in the ti_Flags field of the TaskInfo structure) or if the current state of the task prevents exception delivery (the task must be in the TS_RUN, TS_READY, TS_HIBERNATE, TS_WAIT or TS_WAITAND state, if it is already in TS_EXCEPT, no new exception will be triggered but the already pending one will receive all the exception signals). In these cases, the delivery of the new exception is postponed until allowed again or the previous one completes. When the task is ready for exception delivery, its state is set to TS_EXCEPT (which was already defined in the include files but not used), it is linked into the TaskReady queue and a reschedule is run. When the task is dispatched again, the system will notice the TS_EXCEPT state and call the Exception() Exec private entry point (modified of course by the sysmon.library) via a JMP (original Exec code used a JSR) to avoid leaving a return address on the supervisor stack thus allowing to terminate the task cleanly from exception code. Before calling the routine in tc_ExceptCode in user mode, the exception pending flag (TB_EXCEPT) will be cleared, the exception running flag (TIB_INEXCEPT) will be set and the state of the task before the exception occured will be saved on the task's stack. The signals that triggered the exception are loaded in D0 and cleared in the signal received field and tc_ExceptData is loaded into A1.

Your exception code may return to the normal task code by exiting with a RTS. It is no longer necessary to return the exception signals to reenable in D0 as the new mechanism did not touch the exception masks and thus will make no use of the return code in D0. However, it will not hurt to return them for compatibility reasons. When returning from your exception code, the system will properly restore the task to the state it was in before the exception was triggered, unlike the original Exec code which just left the task running. If the task was waiting or hibernating before and the conditions for waking it up are now met, it will be left running.

If you do not wish to return to normal code, you should call the smEndExcept() function. This call does an implicit smDisallowExcept() to prevent recursion and clears the TIB_INEXCEPT flag. It is thus necessary to call smAllowExcept() after disabling exception signals and before exiting from the program. It is also allowed to call the FreeSignal() or FreeTrap() functions which are normally not permitted inside exception code after the smEndExcept() call.

I have included a trivial example code (ExcptTest.c) in the src directory to show how this is supposed to be used.

You should note that task signal exceptions are definitely an advanced feature that require particular care, in particular when you need to share data between the main program code and the exception routine. You should note in particular that you can not use semaphores in this case as they do only work between different tasks. Instead, you should use smDisallowExcept()/smAllowExcept() in the main program code to protect any code section that should not be interrupted by exception code.

1.73 Version History

CHANGES

V1.20a - 'UnMount ALL' will now ignore ENV: as well as RAM: if mounted as a separate handler. This allows late shutdown code to access global environment variables in all cases and also avoids problems with Env-Handlers that have broken ACTION_DIE support. - Added FRAGS option to the Sysmon monitor Show Memory command

V1.20 - Bumped the replacement alert.hook version to 46 to override the not really new V45 alert.hook in the 3.9 Boing Bag 2 ROM update. - Internal reboot code now clears the VBR and resets the DTT0 registers if appropriate like the V45 exec.library. - Added Show Filesystems command to Sysmon monitor. - Added the LastGuruLog.rexx ARexx script from Christopher 'WatchDog' Elliott as contribution. - Added Unix-like UpTime command.

V1.19 - Duh ! The smExitIntr() system stack checking broke SuperState() (or was it the other way round ?) so SuperState()/UserState() are now replaced to inform smExitIntr() not to bother about the stack while SuperState() is in effect. - Fixed incorrect dump of integer registers for supervisor mode alerts.

V1.18 - Sysmon.library now generates an AN_StackProbe alert if the supervisor stack is invalid in smExitIntr(). This is deactivated by the MACOSKLUDGE option. - Quantum expirations are now counted and the value of the Exec scheduling quantum can be set in the configuration file. - smDispatch() will throw an AN_smInconSchedState alert if a task is found in the ready queue with an inconsistent state. - The Sysmon monitor can now display the exception vectors and the system base structures.

V1.17 - Added a missing Permit() call in the library init routine. It left the ramlib process in forbidden state but had no adverse effect on the system as library init routines are run in forbidden state by exec anyways. - Added the MACOSKLUDGE option as a last effort to support the broken behaviour of MacOS emulators which call Wait() from supervisor mode. This option will disable the AN_smSuperTaskSwitch guru for people who really need to emulate that brain-damaged OS. Updated StartSM and Sysmon.config - The sysmon.library Alert() replacement will now only dump MMU registers if mmu.library is available and reports a working MMU. - The show task command in the Sysmon monitor will now show the tc_MemEntry list.

V1.16 - sysmon.library now has support for load averages and voluntary/involuntary task switches. - sysmon.library smTaskSwitch routine will now allow switching in master mode in an attempt to be friendly to ShapeShifter. An AN_smSuperTaskSwitch guru will still be generated if called in interrupt/supervisor mode. - SysInfo.library will now report load averages and systemwide CPU Usage (total, recent and last sec) and task switches (total and recent, both voluntary and involuntary). - Corrected a cosmetic bug in CPUTime display routines (ShowSys, Timer, Sysmon) which could sometimes show 1000 in milliseconds field due to rounding errors. - Updated the alert database in AlertDump.

V1.15b - SetTrapVectors crashed on the 68000 due to bad CPU check code.

V1.15 First official release of Sysmon V1, revision number not increased since the library is the same than in 1.15 beta. - Timer now also uses THOR's 64/32 integer division routine. - ShowSys now displays the Workbench release number in its banner rather then the KickStart one, this will distinguish disk only releases like 3.5 (V44) or 2.1 (V38). - Sysmon monitor got some additional polish in the show memory command. - Added ExcptTest.c example code to the src directory.

V1.15 beta - The automatic ColdReboot() caching broke on some configurations. It is now a configuration option activated with the new CACHEREBOOT keyword. - The OLDMMURESET option now also applies to software generated resets. This should help softkicked A3000s to reboot without wiping out the KickStart. - Halt REKICK will now clear the Capture and KickTag vectors in ExecBase, wiping out resident modules. - ShowSys and Sysmon now use a 64/32 integer division routine from Thomas Richter to convert EClockVals to regular time. As a side effect of no longer using floating point math, these commands also reduced in size. - The new SetTrapVectors command allows supervisor mode CPU exceptions to also be captured by the extended alert function. This command should be run just before SetPatch in the Startup-Sequence.

V1.14 beta - The library now calls ColdReboot() via a cached vector to bypass patches that may interfere with resets. It needs to be able to reset the system even from supervisor mode or with all tasks frozen. Note that the mmu.library patch is still called as mmu.library will be opened before caching the destination address of the ColdReboot() vector. - smSendBroadcastMsg() didn't set the BMF_DOOMSDAY flag automatically. Fixed. The timeout as also limited to 1/10th second when a reset is pending. - Added OLDMMURESET config option to force reinstallation of the boot MMU config before reboot on a keyboard trapped reset. Use only on systems that fail to reset properly otherwise as sending a KBD_RESETHANDLERDONE command from supervisor mode is a bit dirty. - Updated StartSM, Broadcast and AlertDump commands. AlertDump will now translate the Guru Meditation number. - Added the BcMTest broadcast messages test client. - Added the Show KickTag and Show Broadcast commands to the Sysmon monitor.

V1.13 beta First public beta release of Sysmon V1 - More functions added to the library. - New config options, TRAPRESET, MOVEVBR, RAMLIBPATCH. - New support commands Broadcast, AlertDump. - Rewrote exec task signal exception handling completely. - Added new task scheduling states TS_HIBERNATE, TS_WAITAND. - Rewrote exec Alert() routine, adding a reset resistant logging buffer, reintroducing the 'Guru Meditation' message and using mmu.library if available to properly write the alert data to the zero page. - Added a new alert.hook resident module (V41) to also provide the new display to deferred deadend alerts. - Added a countdown to the shutdown script and a QUICK option for the keyboard reset trap feature. - Added broadcast messages. - smHalt() now uses smFreeze() to stop other tasks instead of RemTask(). - More commands implemented in the Sysmon monitor. - Added an Executive compatible SysInfo.library, with support only for cumulated CPU usage at present. - Documentation updated and converted to AmigaGuide format.

V0.13 - Task switching code was broken on the 68060. Fixed according to Phase 5 documentation for stack frame formats. - Halt REKICK crashed on the 68040 and higher. Fixed. - More commands implemented in the Sysmon monitor.

1.74 Known Bugs or Incompatibilities

Known Bugs or Incompatibilities

It has been reported that ShapeShifter crashes with an AN_smSuperTaskSwitch (C0000007) guru when sysmon.library is loaded. This alert indicates a violation of a very basic AmigaOS programming rule: that task switches are not allowed when the CPU is running in supervisor mode. Sysmon catches this condition early and throws this deadend alert, exec doesn't explicitely check for this but an attempt to Wait() in supervisor mode will quickly result in system data corruption and crash anyways.

I do not use ShapeShifter but I guess that these attempts to Wait() from supervisor mode result from the brain damaged design of the MacOS :-/ and that ShapeShifter probably relies on some hacks or assumptions about the standard exec microkernel behaviour to "just make it work".

Starting with sysmon.library V1.16, the AN_smSuperTaskSwitch alert will only be generated if the CPU is really in interrupt supervisor mode when a call to smTaskSwitch (the internal task switching routine of the sysmon microkernel) is made; an attempt to task switch when the CPU is in master supervisor mode (master bit set in CPU status register) will be accepted. Master mode is not used at all by AmigaOS so Sysmon assumes that whatever uses this mode knows what it is doing.

Starting with sysmon.library V1.17, the MACOSKLUDGE config option can be used to disable the AN_smSuperTaskSwitch alert altogether if you really need it. I still can't guarantee that MacOS emulators will work properly even with this flag set. It's still broken software.

Starting with sysmon.library V1.18, the validity of the supervisor stack is also regularly checked and the system will guru with a deadend AN_StackProbe alert if the system stack is found to be invalid. Checking is done using the SysStkUpper and SysStkLower fields in ExecBase, so hacks that move the system stack without updating these fields may run into trouble. This check is also deactivated if the MACOSKLUDGE option is set.

1.75 Index